Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2018

Fábio Alexandre Henriques da Silva

# Deteção de Ataques de Negação de Serviços Distribuídos na Origem

# Detection of Distributed Denial of Service Attacks at Source

**Universidade de Aveiro**
**2018**

Departamento de Eletrónica,
Telecomunicações e Informática

**Fábio Alexandre Henriques da Silva**

# Deteção de Ataques de Negação de Serviços Distribuídos na Origem

# Detection of Distributed Denial of Service Attacks at Source

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Paulo Jorge Salvador Serra Ferreira, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

**o júri / the jury**

presidente / president

Prof. Doutor André Ventura da Cruz Marnoto Zúquete
Professor auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Mário João Gonçalves Antunes
Professor adjunto do Instituto Politécnico de Leiria (Arguente)

Prof. Doutor Paulo Jorge Salvador Serra Ferreira,
Professor auxiliar da Universidade de Aveiro (Orientador)

**agradecimentos** /
**acknowledgements**

À minha família, por toda a ajuda, carinho e motivação que me deram ao longo deste curso, sem eles não teria conseguido. Ao meu orientador Doutor Paulo Salvador, pelos ensinamentos, orientação, apoio e disponibilidade que sempre demonstrou. A todos os amigos que me apoiaram, contribuindo assim para a elaboração desta dissertação e finalização deste percurso. Um agradecimento especial ao Miguel Oliveira, pelo encorajamento, partilha de conhecimentos, e companheirismo.

**Palavras Chave**

reconhecimento de padões, análise de dinâmicas, modelação de atividade na rede, metodologias para deteção de anomalias, classificação de atividades na rede.

**Resumo**

De ano para ano são estabelecidos novos recordes de quantidade de tráfego num ataque, que demonstram não só a presença constante de ataques de negação de serviço distribuídos, como também a sua evolução, demarcando-se das outras ameaças de rede. A crescente importância da disponibilidade de recursos a par do debate sobre a segurança nos dispositivos e infraestruturas de rede é contínuo, dado o papel preponderante tanto no dominio doméstico como no corporativo. Face à constante ameaça, os sistemas de segurança de rede mais recentes têm vindo a aplicar técnicas de reconhecimento de padrões para inferir, detetar e reagir de forma mais rápida e assertiva. Esta dissertação propõe metodologias para inferir padrões de atividades na rede, tendo por base o seu tráfego: se segue um comportamento previamente definido como normal, ou se existem desvios que levantam suspeitas sobre normalidade da ação na rede. Tudo indica que o futuro dos sistemas de defesa de rede continuará neste sentido, servindo-se não só do crescente aumento da quantidade de tráfego, como também da diversidade de ações, serviços e entidades que refletem padrões distintos contribuindo assim para a deteção de atividades anómalas na rede. As metodologias propõem a recolha de metadados, até á camada de transporte, que seguidamente serão processados pelos algoritmos de aprendizagem automática com o objectivo de classificar a ação subjacente. Pretendendo que o contributo fosse além dos ataques de negação de serviço e do dominio de rede, as metodologias foram descritas de forma tendencialmente genérica, de forma a serem aplicadas noutros cenários de maior ou menos complexidade. No quarto capítulo é apresentada uma prova de conceito com vetores de ataques que marcaram a história e, algumas métricas de avaliação que permitem comparar os diferentes classificadores quanto à sua taxa de sucesso, face às várias atividades na rede e inerentes dinâmicas. Os vários testes mostram flexibilidade, rapidez e precisão dos vários algoritmos de classificação, estabelecendo a fasquia entre os 90 e os 99 por cento.

**Keywords**

patterns recognition, dynamics analyzes, network activity modeling, methodologies for anomalies detection, network activity classification.

**Abstract**

From year to year new records of the amount of traffic in an attack are established, which demonstrate not only the constant presence of distributed denial-of-service attacks, but also its evolution, demarcating itself from the other network threats. The increasing importance of resource availability alongside the security debate on network devices and infrastructures is continuous, given the preponderant role in both the home and corporate domains. In the face of the constant threat, the latest network security systems have been applying pattern recognition techniques to infer, detect, and react more quickly and assertively. This dissertation proposes methodologies to infer network activities patterns, based on their traffic: follows a behavior previously defined as normal, or if there are deviations that raise suspicions about the normality of the action in the network. It seems that the future of network defense systems continues in this direction, not only by increasing amount of traffic, but also by the diversity of actions, services and entities that reflect different patterns, thus contributing to the detection of anomalous activities on the network. The methodologies propose the collection of metadata, up to the transport layer of the osi model, which will then be processed by the machien learning algorithms in order to classify the underlying action. Intending to contribute beyond denial-of-service attacks and the network domain, the methodologies were described in a generic way, in order to be applied in other scenarios of greater or less complexity. The third chapter presents a proof of concept with attack vectors that marked the history and a few evaluation metrics that allows to compare the different classifiers as to their success rate, given the various activities in the network and inherent dynamics. The various tests show flexibility, speed and accuracy of the various classification algorithms, setting the bar between 90 and 99 percent.

# Contents

**Contents** i

**List of Figures** v

**List of Tables** ix

**Glossary** xiii

**1 Introduction** 1

**2 Contextualization and State of Art** 5

2.1 Business Dependence on the Internet . . . . . 5

2.2 Denial-of-Service . . . . . 7

2.3 Distributed Denial-of-Service (DDoS) attacks . . . . . 8

2.3.1 Attack Motivation . . . . . 10

2.3.2 Internet Characteristics . . . . . 11

2.3.3 DDoS Modus Operandi . . . . . 12

2.3.4 Botnet based DDoS attack architecture . . . . . 13

2.3.5 DDoS attacks categories and vectors . . . . . 15

2.3.6 DDoS attack dynamics . . . . . 24

2.4 A glance to the past of DoS and DDoS attacks . . . . . 26

2.5 DDoS Defense Challenges . . . . . 32

2.6 Taxonomy of DDoS Defense Mechanisms . . . . . 33

2.6.1 Based on Approach . . . . . 33

2.6.2 Based on Defense Infrastructure . . . . . 35

2.6.3 Based on Defense Location . . . . . 35

2.6.4 DDoS Defense Goals . . . . . 37

2.7 Source-End Defense . . . . . 38

2.8 Generic Modules of a DDoS Source-End Defense System . . . . . 39

2.9 Machine Learning (ML) in Detection Methods . . . . . 43

2.9.1 Traditional vs Machine Learning Approach . . . 44
2.9.2 Types of Machine Learning Systems . . . 46
2.9.3 Main Challenges of Machine Learning . . . 48
2.10 State of the Art . . . 51
2.10.1 Non-commercial solutions . . . 51
2.10.2 Commercial solutions . . . 52

**3 Methodology for outbound anomalies detection 55**
3.1 Network Data Collection . . . 55
3.2 Features Engineering . . . 57
3.2.1 Packet fields to features . . . 57
3.2.2 Generating network observations . . . 60
3.2.3 Network modeling . . . 63
3.2.4 Features that describe attack behavior . . . 64
3.2.5 Features completeness . . . 64
3.3 High Level Data Overview . . . 65
3.3.1 Dataset analysis . . . 65
3.3.2 Dataset correlations . . . 66
3.4 Data Pipeline and Knowledge Extraction . . . 67
3.4.1 Data splitting . . . 68
3.4.2 Features reduction . . . 69
3.4.3 Features scaling . . . 70
3.4.4 Labeling approaches . . . 71
3.4.5 Outbound anomaly classification . . . 73
3.5 Knowledge Process Evaluation . . . 78
3.5.1 Performance evaluation . . . 78
3.5.2 Zero-day tests . . . 81

**4 Proof of Concept and Evaluation 83**
4.1 Network Data Generation . . . 83
4.1.1 Normal Network Activity . . . 85
4.1.2 Abnormal Network Activity . . . 85
4.2 Parsing packets to metadata . . . 88
4.3 Feature Extraction . . . 89
4.4 Dataset Overview . . . 92
4.5 Classification Methods and Performance . . . 98
4.5.1 Neural Networks . . . 99
4.5.2 Decision Trees . . . 101

4.5.3 Ensemble Methods . . . . . 103

**5 Conclusion and Future Work** **111**

**A Zero-day Extended Results** **115**

A.1 DYN-PW-3 . . . . . 117

A.1.1 DYN-PW-3 applied to ADA12 and GDB18 . . . . . 118

**References** **119**

# List of Figures


2.1 Denial-of-service attack scenario [10] . . . . . 8
2.2 Distributed denial-of-service attack scenario [10] . . . . . 9
2.3 DDoS attacks motivation [23] . . . . . 10
2.4 Agent-Handler Model . . . . . 14
2.5 IRC-Based Model . . . . . 15
2.6 Domain Name System (DNS)-based volumetric reflection attack . . . . . 17
2.7 OSI Model . . . . . 18
2.8 TCP three-way handshake . . . . . 19
2.9 TCP SYN flood attack . . . . . 19
2.10 DNS query flood . . . . . 23
2.11 Representation of previously described attack common dynamics. . . . . 25
2.12 Direct cabling vs Terminal Access Point (TAP) operation mode. . . . . 42
2.13 Traditional approach [129] . . . . . 44
2.14 Machine Learning approach [129] . . . . . 45
2.15 Autonomous Machine Learning approach [129] . . . . . 45

3.1 Metadata collected from the network traffic and how packets fields from different protocols are stacked. . . . . 56
3.2 Partitioning in smaller sampling windows that contain statistical metrics obtained from the metadata during a $\Delta$t time frame. The total number of sampling windows is defined by N = X $\div$ $\Delta$t, X $\geq\Delta$t. . . . . 58
3.3 Low-level structure of a sampling window. . . . . 59
3.4 Second and a proposed third stage, aiming to produce network observations based on the host behavior over a week. The observation windows, at this third stage, have $\Delta$t value of 1 day. . . . . 60
3.5 Observation window horizontal movement throughout the various sampling windows. The shift time is equal to sampling window's length. . . . . 62
3.6 Representation of sliding window paradigm, where at each iteration shifts one minute(i.e., duration of sampling windows), overlapping two windows after the first iteration. . . . . 63

3.7 Representation in the form of histograms of some features obtained from network observations referring to a Generic Routing Encapsulation (GRE) flood attack. X-axis: observed values, Y-axis: count of occurrences. . . . . 66
3.8 *Scatter matrix* for a ack flood dataset composed by three features. Each graph represents the linear correlation between each pair of features . . . . . 67
3.9 General representation of the data pipeline. The data are transformed along the pipeline to be suitable for each ML algorithm. Depending on the nature of these, some steps may be omitted. . . . . 68
3.10 Data splitting applied on two distinct phases. Although split percentages are flexible, in the first split were defined by the *Pareto* principle. . . . . 68
3.11 Representation of *explained variance* according to the number of dimensions [129]. . . . . 70
3.12 Demonstration of Principal Component Analysis (PCA) reduction using Kernel PCA method, and inverse transformation from the resultant projection, maintaining all the relevant information and correlations [156]. . . . . 70
3.13 Representation of the One-versus-All (OvA) strategy, where the classifier responsible for normal traffic produces the highest score, being the new instance classified as normal. . . 72
3.14 Representation of the impact of Support Vector Machine (SVM)'s C parameter separation margin . . . . . 73
3.15 Neural Networks (NN) with an input layer, two hidden layers and a final output layer. When a NN has two or more hidden layers, it is called deep NN . . . . . 74
3.16 Decision tree structure example, with five *leaf nodes* and three *decision nodes*. . . . . 75
3.17 Representation of how the *bagging ensemble methods* operate on the training set, dividing it into several that feed the same number of predictors that ultimately unify, through a voting process. . . . . 76
3.18 Representation of the *boosting ensemble methods* iterative process, intending to adjust the weight of an observation based on the last classification. . . . . 77
3.19 *Confusion matrix* for a binary classifier. The paradigm can be transposed to a multiclass classifier, where the matrix grows proportionally either in rows and columns, maintaining the correct classified instances on the diagonal from top left to bottom right. . . . . 79
3.20 Receiver Operating Characteristic (ROC) curve for SVM multi-classification using Wine dataset [162]. . . . . 80

4.1 Enterprise hierarchical network layers . . . . . 84
4.2 Representation of both dynamics. . . . . 86
4.3 The three equal parts represents the 12 means, medians and variances that result from the application of each statistical operation to each attribute. The last two relate to the number of periods of silence and their average duration. Thus, each observation window contemplates this set of 44 features. . . . . 92

4.4 Number of captured packets per attack vector including also normal network activities. . 93
4.5 Number of generated network observations per attack vector, including also normal network activities . . . . . 94
4.6 Histograms of some features present on observations obtained from an ACK flood scenario. 96
4.7 Representation of *explained variance*, the smaller the number of features the smaller the variance ratio and therefore the greater the difference between what the pattern expresses in relation to the original. . . . . 97
4.8 Representation of *explained variance* in function to the number of features, for each dataset. 98
4.9 *Learning curve* for *ANN4*, showing how learning improves with samples number. . . . . 100
4.10 Normalized *confusion matrix* for *models 10* and *11* over the test set(i.e., 20 % of dataset *DT-4* and *DT-5* respectively). . . . . 102
4.11 *Learning curve* for *DTC11* . . . . . 103
4.12 On the left *Adaboost* learning curve after *cross-validation* over the *DT-1* 80 % while on the left there is represented the ROC curve computed from the classification on the test set. . 104
4.13 Distribution example of the duration of each period of silence resulting from the exponential distribution with scalar equal to 0.1. . . . . 105
4.14 Example of the scale parameter variation over the resultant silence periods duration, being high stealth when assumes lower values. . . . . 107

# List of Tables


3.1 Relation between observation window's length and number of observations generated. It follows the previous example, where sampling windows assumed a $\Delta$t of one minute and the capture total length is one week indexed on 10080 sampling windows. . . . . 61

3.2 Number of observations produced from varying observation window's size and sliding window shift time. Values are also based on the previous example. . . . . 62

3.3 *Pulse-wave* dynamics that vary the maximum number of packets as well as the periods of silences, which after observation for two minutes result in the total number of packets represented in the third column. For the sake of simplicity, it was assumed that the packet throughput is instantaneous. . . . . 81

4.1 Dynamics used during the acquisition of attack data, the first being more intrusive and completely immutable, while the others assume some parameters in order to vary the degree of intrusiveness. . . . . 87

4.2 Pool of attack vectors and respective alias . . . . . 87

4.3 Summarizing the contents of each dataset, considering the various dynamics and simultaneity with normal services, as described in Section 4.1.2. The presence of network observation from normal activities is common in all datasets. . . . . 89

4.4 Low level description of the attributes present on every sampling window. . . . . 90

4.5 View of a random portion of samples that make up the total dataset. Each column represents a feature while each row is a sample. Should be noted that the first column is a simple identifier, and all the features present in the table relate to outgoing traffic. . . . . 95

4.6 Number of components to kept for each dataset. . . . . 98

4.7 Overall NN classification performance over 10 validation sets, for each dataset with respective optimal features number for each one. . . . . 99

4.8 *Zero-day* test performance for each model with respective ideal number of components. Since this is a one-time classification, it is not possible to provide a confidence interval. . 101

4.9 *Decision tree* classification performance for each dataset, with respective optimal number of features. . . . . 101

4.10 *Adaboost*, on the left, and *Gradient-boost*, on the right, classification performances, maintaining the dataset and PCA for both algorithms. . . . . 104
4.11 Adaboost and Gradient-boost classification performance for two attack vector, both with same dynamic. Extensive results available in Table A.1 and A.2. . . . . 106
4.12 Adaboost classification performance including dissociation of patterns in Youtube traffic, for two attack vectors. Extended results are present in Table A.4 and A.5. . . . . 106
4.13 *Gradient-boost* and *Adaboost* performances comparison, from left to the right, maintaining the *zero-day* samples per row. Extended results are provided in Section A.1. . . . . 107
4.14 Comparison between the results obtained using the same *zero-day* tests over two different ML algorithms, on the left *Gradient-boost* and on the right *Adaboost*. Extended results available in Section A.1.1. . . . . 108
4.15 *Adaboost* model ADA12 results for low-rate ACK flood. . . . . 108
4.16 *Gradient-boost* model GDB18 results for low-rate DNS-query flood. . . . . 109

A.1 *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., Transmission Control Protocol (TCP)-SYN flood) action following *DYN-PW-1*, which have a peak of 300 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1. . . . . 115
A.2 *Gradient-boost*(i.e., *GDB22*) and *Adaboost*(i.e., *ADA16*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-1*, which have a peak of 300 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1. . . . . 115
A.3 *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following a dynamic with a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1. . . . . 116
A.4 *Adaboost*(i.e., *ADA13*) classification performances over samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-1*, which have a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. The action of the attack vector was initially captured simultaneously with Youtube activity, and later in isolation. . . . . 116
A.5 *Gradient-boost*(i.e., *GDB19*) classification performances over samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-1*, which have a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. The action of the attack vector was initially captured simultaneously with Youtube activity, and later in isolation. . . . . 116

A.6 *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 117
A.7 *Gradient-boost*(i.e., *GDB20*) and *Adaboost*(i.e., *ADA14*) classification performances over the same samples obtained from the *AV-2*(i.e., TCP-ACK flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 117
A.8 *Gradient-boost*(i.e., *GDB21*) and *Adaboost*(i.e., *ADA15*) classification performances over the same samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 117
A.9 *Gradient-boost*(i.e., *GDB22*) and *Adaboost*(i.e., *ADA16*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 117
A.10 *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 118
A.11 *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-2*(i.e., TCP-ACK flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 118
A.12 *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 118
A.13 *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. . . . . 118

# Glossary

| | |
|---|---|
| **ABIDS** | Anomaly-based Intrusion Detection System |
| **API** | Application Programming Interface |
| **AUC** | Area under the curve |
| **AI** | Artifficial Intelligence |
| **CPU** | Central Process Unit |
| **CEO** | Chief Executive Officer |
| **C&C** | Command and Control |
| **CERT/CC** | Computer Emergency Response Team Coordination Center |
| **DM** | Data Mining |
| **DPI** | Deep Packet Inspection |
| **DoS** | Denial-of-Service |
| **DDoS** | Distributed Denial-of-Service |
| **DNS** | Domain Name System |
| **FPR** | False Positive Rate |
| **FTP** | File Transfer Protocol |
| **GMM** | Gaussian Mixture Model |
| **GRE** | Generic Routing Encapsulation |
| **GUI** | Graphical User Interface |
| **HMM** | Hidden Markov Model |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Secure Hypertext Transfer Protocol (HTTP) |
| **ICO** | Initial Coin Offering |
| **ICMP** | Internet Control Message Protocol |
| **IoT** | Internet of Things |
| **IP** | Internet Protocol |
| **IRC** | Internet Relay Chat |
| **ISP** | Internet Service Providers |
| **IPX** | Internetwork Packet Exchange |
| **IDS** | Intrusion Detection System |
| **IPS** | Intrusion Prevention System |
| **ML** | Machine Learning |
| **MLP** | Multi-Layer Perceptron |
| **NMS** | Network Management System |
| **NN** | Neural Networks |
| **OvA** | One-versus-All |
| **OvO** | One-versus-One |
| **OSN** | Online Social Network |
| **OSI** | Open Systems Interconnection |
| **OS** | Operating System |
| **OCR** | Optical Character Recognition |
| **PBFS** | Packet-Based per Flow State |
| **PCA** | Principal Component Analysis |
| **POC** | Proof of Concept |
| **QoS** | Quality of Service |
| **RAM** | Random-access memory |
| **ROC** | Receiver Operating Characteristic |
| **SPI** | Shallow Packet Inspection |
| **SPoF** | Single Point of Failure |
| **SMTP** | Simple Mail Transfer Protocol |
| **SNMP** | Simple Network Management Protocol |
| **STOMP** | Simple (or Streaming) Text Oriented Message Protocol |
| **SaaS** | Software as a service |
| **SVM** | Support Vector Machine |
| **SPAN** | Switch Port Analyzer |
| **TAP** | Terminal Access Point |
| **TTL** | Time to live |
| **TCP** | Transmission Control Protocol |
| **TPR** | True Positive Rate |
| **UDP** | User Datagram Protocol |

CHAPTER 1

# Introduction

When one thinks about a typical corporate organization structure, the physical layer, involving not only the network topology but also the surrounding built environment, and social domain comes quickly to mind. Usually, both physical and social domains follow an explicit hierarchy, where each layer or constituent, respectively, has a set of well defined objectives. Those when aggregated, allow the enterprise to reach their goals, ensuring the correct execution of the internal operations, allowing to maximize the availability of services that it exposes. Following the pyramid hierarchy of the social stratum, the greater the distance to the peak is, the fewer responsibilities are associated with the elements that characterize it, and the more unpredictable and susceptible to failure are. On the contrary, at the physical level, including constructions and equipment of the various sectors, they are highly inflexible, and arranged according to a set of rigorous criteria and policies. These aim to make the most of it without compromising the safety and conscience of all the subdivisions of the physical hierarchy. Typically, with the increased business scale the greater the exposure of the corporation and invariably more rules need to be applied in both social and physical domain. However, it is increasingly difficult to manage the broad base of the social domain pyramid, not only at the access control level but also at secrecy and activities within the corporate environment.

Transversal to all industries in the tertiary sector of the economy, and independently of the protocols and management of policies that are typically empowered in the physical layer to fill the gaps inherent in the conjugation of the human with the machine, the social domain continues to be the most conducive to subversion. Appealing to cognitive bias through notable techniques of social engineering, the presence of anomalies, sooner or later will be reflected in the orchestration of physical resources. Tipically this leads to the divergence from the general workflow of the company and subsequent scope and propensity to capitalize. Human behavior represents the most exploited point of failure and also constitutes the access layer for a set of threatening attack vectors which compromise part of the corporation's physical structure and sustainability. Different companies are characterized by distinct risks and threats inherent in either market competition, poor planning and management of physical domains, or even by

adapting to current technologies that provide long-term support. These digital assets have evolved so fast that they now represent a preponderant role in any company. Since they simplify the work in such a way that allows its execution in a faster, intelligent and more secure way providing tangible and intangible benefits. Nonetheless, technology is heavily reliant on the internet to deliver all of its features to the fullest, which is also very prevalent in corporate environments. Many businesses trust in these two areas their scope subsequent income. Nowadays with cloud advent, it is very difficult to imagine how any business can operate and thrive without the use of the Internet. The digital interconnection of the world is discussed in Section 2.1, along with the dominance of the Internet in the business circle, also highlighting the influence of the resources availability.

Given the constant growth of the power of digital devices and their preponderance mainly in the labor environment, technological advance means changes and opportunities, which always assume a dubious result. Such changes may reflect improvements in well-being in general, or highly abused against the availability of some service or infrastructure, as discussed in Section 2.3. This type of offensive is generally punished by justice, because the motivations are mostly driven by bad faith and the intention to cause damage. However in some cases, despite the inherent consequences of these incidents, are used as a form of protest and expression of public opinion, as detailed in Section 2.3.1. Furthermore, is described the categories and implicit attack vectors that make this topic so current and frequent sustained by the history and most important milestones in Section 2.4.

Highlighting the importance of such threats, it is mandatory to report the set of requirements and modules present in any defense system, that broadly aims at increasing the security, integrity and availability of a network. With this in mind, the focus is shifted to the topic of machine learning, discussed in Section 2.9. It is excessively dissected these days [1], largely because of the large amount of data present in any sector, which has contributed to the remarkable advances. These are particularly great in image and language recognition, which in general allow them to surpass human levels of comprehension. This area, when integrated in scope of detection, has the margin to raise the bar of detectability and awareness to traditionally unreachable levels.

The work developed on this dissertation proposes a source-end solution, based on a set of methodologies, strategies and requirements, which tries to detect corporate network anomalies. The advantages of machine learning techniques strongly encourage its use, allowing to infer subtle patterns of activities and services present in the business environment, identifying their normality or abnormality, according to typical behavior. By evaluating activities in the past, allows to identify patterns in the future. Based on this paradigm, it is possible to raise the level of awareness of the internal security of the corporate network, and maintain the normal workflow. This approach prevents the channeling resources and operations to areas that do not aim at productivity and revenue. The work developed in this dissertation includes variations of the stealth degree of several attack vectors used in the recent past. Analyzing in depth the flexibility of the solution, as mentioned in Section 3.5.2, allows to delineate a detectability range of anomalous behaviors.

This dissertation was developed at *IT - Institute of Telecommunications* at University de Aveiro, where the workspace and all the necessary conditions were kindly provided.

CHAPTER 2

# Contextualization and State of Art

*Corporations have increasingly relied on the Internet to provide the one set of services. This dependency creates an opportunity on a crucial point of any business as it typically holds the largest source of income. Usually, current corporate environments focus excessively on defense against external threats, devaluing the awareness of the internal network, which in the presence of anomalies drives the resources inherent to the business against the availability of the services of another corporate entity. In this chapter, an overview of the currently distributed denial of service attacks is presented, along with the consequences on the economy and reputation of all the stakeholders. It also reviews some requirements on the development of defense solutions, as well as the use of machine learning techniques given the increasing amounts of monitoring data, extends the margin of progression of these systems, providing levels of reliability and detectability hitherto unattainable.*

## 2.1 Business Dependence on the Internet

The new era of the 21$^{st}$ century is dominated by exponential growth of connectivity in the world. Following a bottom-up approach, everything is connected to the Internet, from the pocket, through homes, to corporations. Mobile as 7$^{th}$ mass media, nowadays, guarantees access to applications, services and multimedia content, mainly through an Internet connection. The dependence on network operators no longer prevails, and currently mobile devices represent only a means of access to the Internet, and to all the services that it makes available. Shifting the focus to the home environment, there is a step growth of Internet of Things (IoT) devices, such as televisions, home cameras, sensors, which, in order to provide all of their functionality, also depend on an internet connection. Owing to this advance of the technology and Internet improvements, corporations felt the need to keep up with the era and create dependence on the Internet as a way to provide their core business operations and to reach every corner of the world economy. Yet, despite the openness of the network, connectivity, and the ease of providing services over the internet, it can also be a double-edged sword creating new and unfamiliar vulnerabilities to enterprise business processes. Increasingly, companies are

aware of the risks and threats inherent in the increasing reliance on the Internet. In addition to tackling the gaps in Internet design and protecting the operations of various adversities, most companies tend to implement security policies at a variety of levels, from security to physical level, to the personal level, to the level of communication and network up to the level of information. To ensure business continuity, these last two layers are the most important. Security at the network level is a process designed to protect the usability and integrity of the corporate network and data. On the other hand, information security[1] is a set of practices that aim at a balance between a triangulation consisting of three fundamental security concepts: confidentiality, integrity and availability(CIA)[2]. Most of the organizations follow these to guide policies for information and services security within them. The confidentiality relies on the principle of "least privilege" which states that only the minimum access necessary should be granted for the least possible amount of time to perform an operation [2]. Thus, confidentiality is a component of privacy which can be implemented by limiting the access to a critical service or sensitive information allowing only the access by authorized entities. Integrity is concerned with the maintaining of consistency and trustworthiness of data over its entire lifecycle. This concept ensures that the data must not be changed whenever is in transit or stored [3]. It also prevents unauthorized users from making intentional pr accidental improper changes compromising the information integrity along with subsequently value and veracity. The last face of the triangle - availability - ensures that information and resources must be accessible to those who are authorized and need them.

This crucial concept is the most comprehensive of all, and can be transposed to another layer. More than ever, companies are converging to a point of total dependence between the availability of their resources and the ability to monetize, protect revenue, ensure employee productivity, and provide an excellent end-user experience. With the advent of cloud computing there is a worldwide influence to shift services to the cloud, supported by the various benefits it offers to organizations, mostly economic. For instance, Google Docs is one of the Software as a service (SaaS) that Google provides, where there is a total reliance on the software's availability in the cloud and the internet infrastructure's availability. Similarly to a landlord who rents an apartment, tenants depend on the elevator's availability and stairs to access the apartment, illustrating the service provided, but also the availability of access to the building in general, exemplifying the internet. Thus, providing solid and uninterrupted services strengthens customer relationships, while in addition to increasing trust and loyalty to them, it increases its satisfaction and therefore the good name's proliferation and corporation's reputation. Hereupon, companies that rely on these availability-sensitive services are constantly threatened by cyber criminals executing availability targeted attacks [4]. Such attacks are often named Denial-of-Service (DoS).

[1] Also known as InfoSec

[2] Also referred as AIC triad

## 2.2 Denial-of-Service

Usually, the concepts of denial-of-service and denial-of-service attacks are wrongly considered as equal. They are two completely different concepts where the first one refers to an event or situation and the second refers to an intent-driven illicit act. However, this intent may be the denial-of-service and, following this way, they share a causal relationship, where the former comes as an effect caused by the latter. According to [5], a "Denial-of-Service (DoS) is an event or a situation, in which a legitimate client cannot access the requested service to which the client is entitled to and which should be available". In its turn, a DoS attack is a malicious attempt by an attacker to disrupt online services of a service provider (server) and to make it unavailable to its legitimate users (clients), as reported by [6]. In addition to the relationship, this is actually, an availability problem in a client/server environment, where the number of targets or attackers are irrelevant once the single purpose of te attack must be to cause a denial-of-service. However, there are other attacks that may cause DoS situations as a side effect which we can't conclude about its intentionality, hence, characterize what can be or not a DoS attack. For instance, due to the explosive growth of the cryptocurrency, bitcoin and everything associated with it has hit peak commercial popularity which led to the emergence of countless cryptocurrency mining virus [7].

These malwares(e.g., *WannaMine* [8]) are designed to mine a specific cryptocurrency adding the respective currency to a digital wallet belonging to their creators. The underlying mining process is carried out by trying to make maximum use of the Central Process Unit (CPU) and Random-access memory (RAM) placing the computer under great strain as well as trying to spread across the network which can cause a DoS situation. Thus, these attacks cannot be characterized as DoS attack once the primary functions of cryptocurrency malwares are to mine and propagate. It is entirely plausible to assume that this kind of malicious software may led to DoS incidents, but there must be a clear conceptual separation between true DoS attacks and side effects of other attacks.

A deliberate DoS attack occurs when the victim receives a malicious stream of packets, sent by an attacker which has as objective to exhaust some key resource, denying its service to the victim's legitimate clients [9]. Figure 2.1 depicts a typical denial-of-service attack scenario in which a single attacking machine A sends a set of consecutive packets to victim V, preventing them from providing their service to legitimate clients C1 and C2. Usually, attacker don't use their own machines to perform attacks, increasing the difficulty to traceback and avoiding being discovered. So machine A is, in this scenario, an agent machine, a device compromised by the attacker who involuntarily participates in these attacks.

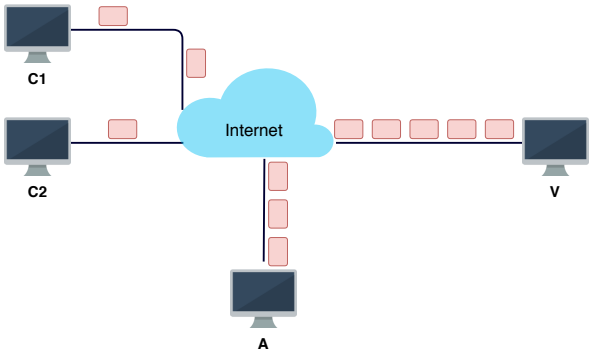



**Figure 2.1:** Denial-of-service attack scenario [10]

The victim resource could be exhausted performing an attack by exploiting some vulnerability in the software running on the victim side(*vulnerability attacks*) or by simply sending a huge amount of unexpected traffic which victim cant handle(*flooding attacks*). No matter the attack vector and its exhaust method, the effect will be the same, and in the absence of effective defense mechanism, it lasts for the entire duration of the attack. Once the attack is aborted, if there is no hardware constraints as result of the attack, the resource should be able to recover, coming back to be available for those who need it. However, depending on the attack vector, the denial-of-service effects can range from slight increases in service response time to complete inaccessibility which commonly provoke financial implications on the organizations that are heavily reliant on the availability of their service. For example, Amazon [11] calculated that a page load slowdown of just one second could cost it $1.6 billion in sales each year [12]. So, if a minimal slowdown can cause a loss of revenue with this dimension, we can infer a direct and exponential relationship between availability and impact in the financial sector of corporations and that's why DoS incidents are a major concern to enterprises nowadays.

## 2.3 Distributed Denial-of-Service (DDoS) attacks

Mirkovic et al. [9] defined a DDoS attack as an association of multiple machines, each one deploying a DoS attack towards one or more targets. However, this definition fails to include the aspect of coordination between the attacking hosts, which is one of the three fundamental characteristics of a DDoS attack. The second feature is a requirement in that must exist more than one source serving as attack platform, to perform it. Likewise, this platform is composed of a set of agents machines which are compromised devices that unwitting provide its resources. Otherwise, without this platform, it would not be categorized as a DDoS attack but rather as a DoS that refers to the third characteristic: DDoS attacks are a subset of DoS attacks. It inherits the motivation, some features and amplifies the effectiveness of DoS significantly, increasing also the complexity of defense problems [13]. Briefly, a distributed denial-of-service attack is large-scale, coordinated attack on the availability of services of a victim system or network resource, launched indirectly through many compromised devices on the Internet, as stated in [14], [15].

It is important to notice that, rather than the concept of denial-of-service there is no notion of distributed denial-of-service, once the service, as a whole, can be denied and therefore cannot be distributively denied, unless it follows a distributed architecture. There must only exist the concept of distributed denial-of-dervice attack.

Figure 2.2 shows the most frequently used scenario where the attack platform is constituted by two machines, A and B, which simultaneously starts generating a set of consecutive packets to victim V, preventing them from providing their service to legitimate clients C1 and C2.

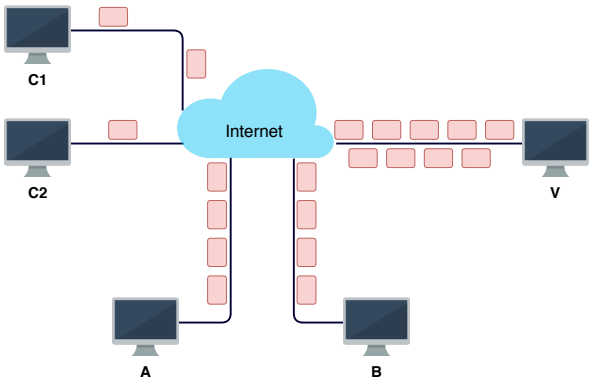



**Figure 2.2:** Distributed denial-of-service attack scenario [10]

Despite the different scenarios, there are three important features that characterize attack of this nature:

- **Use of Internet Protocol (IP) source spoofing**

  Generally, attackers use source address spoofing during the attack by forging the information in IP source address field in attack packet headers. This is a common practice that is cross-sectional to DoS and DDoS attacks with a set of advantages associated with each. One of the benefits that attackers receive from IP spoofing is that it makes extremely difficult to trace the agent machines. Thus, since there is a very low risk of being traced, stored information(i.e., access logs) cannot help to locate the attacker himself or even facilitates this process. This is a major fact that encourages substantially the occurrence of denial-of-service incidents once enables the attacking party to avoid accountability for the attack [16]. Besides that, hiding the identity of agent machines enables the attacker to reuse them for further offenses. Furthermore, in DDoS scenarios, once there are a significant number of devices each one producing packets with respective fake source IP address they appear as if they come from disparate sources. Thus, straight-forward solution to resource overloading problems such as fair-sharing techniques is ineffective against DDoS attacks, given the vast number of IP addresses. The other benefit that IP spoofing allows to the attackers use attack vectors even more sophisticated(i.e., *reflector attacks*).

- **Similarity of attack to legitimate traffic**

  Attackers tend to obscure the malicious flow within legitimate traffic by generating legitimate-like packets. Besides that, they also use the patterns of flash crowds[3] causing

[3]Legitimate user's traffics

attack traffic to be perceived as legitimate user's traffic [17]. Thus, since malicious packets do not stand out from legitimate ones, it is extremely hard to sieve legitimate from attack traffic based purely on examination of individual packets. In order to differentiate traffic, a profile of the legitimate one must be created through inferred patterns(e.g.,statistical patterns) that when intersected with attack traffic allows finding deviations and correlations which demarcate DDoS attacks from flash crowds [18].

- **Wide number of agent machines** Typically, the effectiveness of an attack can be predicted according to its number of agent machines, since there is a direct proportionality between the two variables. There is a steady increase in the ability for attackers to easily deploy large DDoS attack networks, which nowadays, continue to outpace available bandwidth in most cases[19], allowing the victim's resources to be overloaded, making the attack effective. In addition, with a large number of agents, even if traceback could be successfully performed in the face of IP spoofing, it is difficult to take a decision against thousands of agents. Thus, a large number prevents any automated responses aimed at stopping attack flows between the source and target, since even if a agent machine is disabled there are hundreds or thousands, that can proceed with the attack.

### 2.3.1 Attack Motivation

DDoS attackers are usually motivated by various reasons. Historically, ideological hacktivism [20] has commonly been the top motivation, however, things have changed. Actually, according to Arbor Reports [21]–[23] for two consecutive years, gaming is the leading impetus. With a slight difference of 1.4% arises the criminals which demonstrate its attacks capabilities and its attack vectors pool followed by criminal extortion attempts rounding out the top three motivations, as illustrated in Figure 2.3.

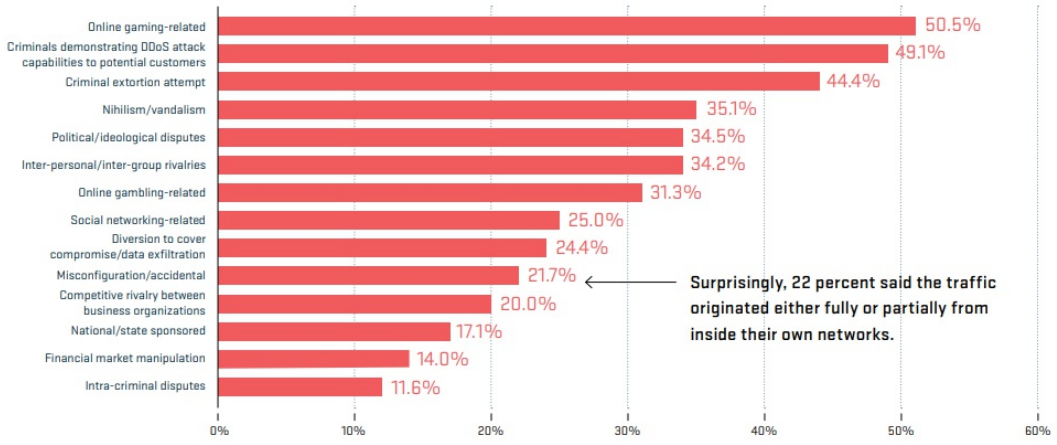



**Figure 2.3:** DDoS attacks motivation [23]

However, despite the smaller percentage are also important attacks motivated by nihilism/vandalism that according to the same source has been at the top for the last 5 years [24]. It is also important to note that some DDoS attacks are being used as a distraction, acting like a smokescreen [25], for either malware infiltration or data exfiltration and that this

method has been growing significantly. The fact that one of the main motivation is related to the intellectual challenge is worrisome in that it is an indicator of the ease with which DDoS attacks can be acquired and executed increasing the seriousness of the availability's problem of Booters/Stressers[4] services [26].

### 2.3.2 Internet Characteristics

Typically, the attacking machines are geographically-distributed connected by the Internet which is the mainly reason that makes the DDoS attacks so powerful. They take advantage of the Internet architecture, which it rests on a principle that aims at functionality at the expense of security. It provides fast, simple and cheap mechanisms enforced with various higher-level protocols that ensure reliable timely delivery of messages with certain level of Quality of Service (QoS).

Internet design follows two principles: best-effort service and end-to-end paradigm. The first one enable the conception of numerous transport protocols on top of the IP to provide various performance guarantees(e.g., Transmission Control Protocol (TCP) for reliable delivery) [27]. The last one enables end users to manage their communication according to their's desired service guarantees, allowing to add complexity and functionalities on the Internet edges, by using new protocols while the intermediate network remains simple and efficient and it only needs to deliver IP packets without needing to understand services above the network layer. In this context, problems arise when one of the endpoints becomes malicious and acts to damage the others endpoints violating the end-to-end protocols and providing no more guarantees. The end-to-end paradigm allows to passively continue forward packets to the destinations, overwhelming the victim's resources, once it prevents the intermediary network from policing or taking action on traffic from the malicious endpoint. The Internet design opens several security issues and creates opportunities for denial-of-service incidents [15], [28]:

- **Internet security is highly interdependent**

  DDoS attacks are commonly launched from several points on the Internet that are external to the victim's own system or network which were subverted through security-related compromises. Regardless of victim's security system, it is always susceptible to DDoS attacks as it depends on the state of security in the rest of the global Internet[29].

- **Internet resources are limited and the power of many is greater than the power of few**

  Each Internet entity (e.g., host, network, service) has limited resources(e.g., bandwidth, processing power, storage capacities) that can be consumed by too many users. Thus, in the absence of defense mechanisms, every DDoS attack with a sufficiently large pool of agents will always be detrimental if its resources are greater than victim's those.

- **Intelligence and resources asymmetry**

  As a consequence of the end-to-end communication paradigm, all logic and complexity are located at endpoints allowing to limit the amount of processing in the intermediate

[4] Also known as DDoS-for-hire services

network so that packets could be forwarded quickly and at minimal cost. To accomplish that, intermediate network needs high bandwidth pathways while edge networks, once it provides services to fewer customers, only invest in bandwidth according to its necessities. Thus, malicious clients can misuse the bountiful resources on the intermediate network to delivery a huge amount of messages to a victim.

- **Accountability is not enforced**

  As stated in 2.3, it is erroneous assume that the source address field in an IP packet is the IP address of the machine that originates the packet. This gives attackers a powerful mechanism to escape accountability for their actions[30].

- **Decentralized Internet Management**

  As described in [31], the Internet is an aggregation of numerous networks which are interconnected and assumes a behavior according to its local policies by its owners. Due to the lack of central control, there is no way to enforce global deployment of a particular security mechanism or policy, and due to privacy and other commercial concerns network service providers generally are reticent to provide detailed information about the traffic patterns within their networks.

### 2.3.3 DDoS Modus Operandi

A distributed denial-of-service is carried out in several phases. These stages follow a pipeline architecture, having a high dependency between the phases. Regardless of the agent machines's architecture, it is possible to enumerate 4 generic phases:

1. **Selection of agents**

   The attacker looks for machines that it can compromise. To maximize the yield, there are some requirements that play an important role in the process of choosing machines. Usually, the attackers tend to victimize devices which have good connectivity, abundant resources and are poorly maintained, so that a powerful attack stream can be generated. Typically, in the early years, the attempt to acquire control of agents machines were a manually process, however, with the development of advanced scanning tools this process has become automatic, easier and immediate [31].

2. **Compromise**

   To compromise the selected agents, the attacker exploits security holes and vulnerabilities of the selected agents and deploy the attack code. Nowadays, often this stage is merged with the above one, where a single program[5] assume 3 distinct primary functions [32]: scanning, exploitation and deployment, executed by the respective order. Firstly the program needs to look for vulnerable devices on a specific network. After identifying them, you must commit them and establish a connection with a Command and Control (C&C) server reporting the acquisition of a new agent. Finally, the program must deploy the malicious code.

   Despite this life cycle, these sophisticated types of software contain also automated tools for covering tracks, preventing the identification and deactivation by erasing all

[5] Also known as worm

the logs which show malicious activity and destroying any others evidence that could incriminate the attacker.

3. **Communication**

   In order to initiate the attack, the attacker needs to communicate with each agent to conclude about its availability to be active part of the attack. This communication can be via various protocols such as Internet Control Message Protocol (ICMP), TCP, User Datagram Protocol (UDP) or Hypertext Transfer Protocol (HTTP)/Secure Hypertext Transfer Protocol (HTTP) (HTTPS).

4. **Attack**

   In the absence of a schedule on the attack source code, the attacker needs to communicate with each agent in order to initiate the malicious action by setting up the last details such as the selected attack vector, IP's, port numbers, Time to live (TTL), etc.

The scope of this dissertation focuses only on the last phase. The scenario is at the organizational level, but it is assumed that the internal device is already infected, compromised, and carries a set of attack vectors prepared to overthrow an outside entity.

### 2.3.4 Botnet based DDoS attack architecture

Often, the agents machines[6] presented on every DDoS attack share a relation between them once they are controlled, directly or indirectly, by an attacker, making use of agent's modest capabilities to overload the victim's resources thus contributing to their goal. This operation mode gave rise to the concept of *botnet*, which compromises a network of machines with malicious programs, implement under a Command and Control (C&C) management infrastructure and carry out the attack. Similarly, botnets derive their power by scale, both in their bandwidth, reach and in their ability to cause disruption [33]. Botnets are not exclusive to distributed denial-of-service attacks, they can carry other types of attack, or simply make use of participants resources to perform malicious activities such as cryptocurrency mining as recently reported by [34]. These collections of bots spread over the Internet are getting larger at a high rate, and according to the logic associated with each slave, it can assume different functions on the botnet architecture. Botnet based DDoS attacks are generally launched using three basic models:

- **Agent-Handler Model**

  In this model, as shown in Figure 2.4, there are four participants: (i) the master or the attacker, (ii) the handlers, (iii) the agents and the (iv) victim. The first entity, i.e., the attacker initially attempt to compromises some devices in a network to bring them under his control. The handlers, in turn, include some malicious software residing on remote machines that are used by the attacker providing an indirect means of communication to command and control the agents . As stated before, it is common for an attacker to launch a DDoS attack from the handlers creating one or more abstraction

[6] Also known as bots, zombies, slaves, agents, demons

layers which prevents to trace the attack back to the attacker. The agents are a set of compromised machines that are responsible for performing the attack. Finally, the victim is the final endpoint or it may be a group of target machines [35]. The operating mode is similar to that described in Section 2.3.3 where the attacker communicates with any of the handlers to identify operational agents and schedule the attack. Depending on the configuration of the DDoS attack network, agents can be instructed to communicate with one or multiple handlers in order to facilitates the commands proliferation process also preventing the existence of Single Point of Failure (SPoF).

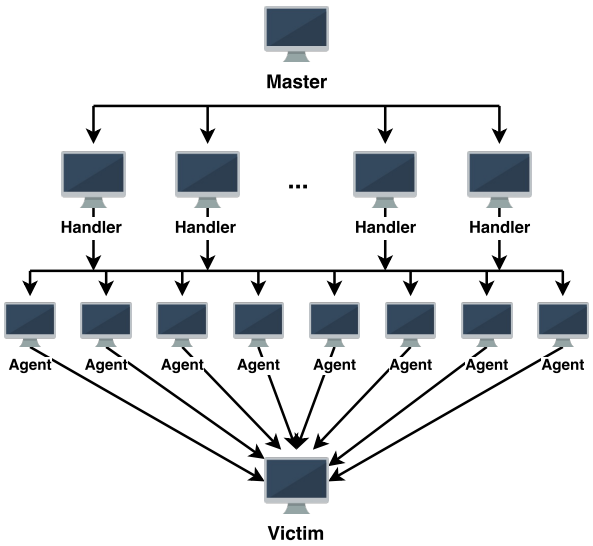



**Figure 2.4:** Agent-Handler Model

- **Internet Relay Chat (IRC)-Based Model**

  The architectures of the IRC-based DDoS attack as shown in Figure 2.5 and of the agent–handler model are almost similar. However, instead of the communication between the attacker/master and the agents being mediated through a layer of handlers, it is carried out through an IRC communication channel. The internet relay chat is a text-based chat system that organizes communication in public channels allowing users to communicate without performing any authentication. Thus, once installed in the agents, the bot will automatically join a specific IRC channel on an IRC server, and wait for further instructions [36]. Although it is a simple process, with low latency and provide communication advantages as well as the identification speed of available agents and the usage of legitimate ports, it is not a scalable architecture and creates a SPoF since it has a centralized C&C model [37].

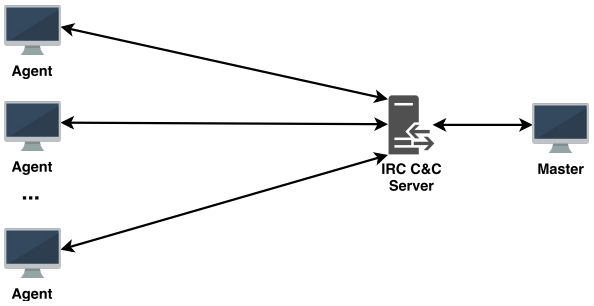



**Figure 2.5:** IRC-Based Model

- **Web-Based Model**

  In the past years, web-based models have emerged and rapidly were adopted as preferred platform for botnet command and control. A web-based botnet is a botnet whose C&C server and bots use the most universal and supported network protocol i.e., HTTP, to communicate with each other. The C&C procedure is simple, requiring only the existence of a web server, where each agent acts as a web client, making mostly GET and POST requests. Due to HTTP offers many advantages since, nowadays, the HTTP traffic is the most popular Internet traffic so that botnet traffic can be easily concealed within normal HTTP traffic, making these botnets more difficult to be discovered in relation to the ones that use other protocols less used today. Generally, most network firewalls/proxies allow hosts behind them to access Internet via the HTTP and HTTP over SSL/TLS[7] using legitimate ports: 80 and 443 respectively [38]. Thus, web-based botnets can easily provide stable, qualified and scalable client-to-server communication model(e.g., Spyeye [39], Zeus [40]). Despite all the advantages, with the abrupt growth of Online Social Network (OSN) [41] and the ease of creating fake profiles without an associated cost, attackers tend to use these platforms as a means of communication to command and control agents which usually, just need to visit periodically a specific profile on a social network(e.g., Facebook, Twitter, Tumblr), parse the HTML page and process the posted commands [42], [43].

### 2.3.5 DDoS attacks categories and vectors

DDoS attack vectors vary significantly, and cybercriminals are constantly using a number of different techniques, evolving their methodologies to implement DDoS attacks that disable or overload the target business's IT infrastructure by evading the recent defense mechanisms. Despite the various taxonomies of DDoS attacks [9], [28], [44] and its inherit metrics like the degree of automation, exploited vulnerability, source address validation, possibility of characterization, attack rate dynamics, impact on victim, victim type, etc, it is possible to classify DDoS attacks based on the type and quantity of traffic used for the attack and the exploited target's vulnerability. Keeping present that all the categories are botnet-based, it is

[7] Also known as HTTPS

possible to obtain 3 groups that are already widely profiled and accepted as standard [21], [45]–[47]:

- **Volume-Based Attacks**

  In volume-based (or volumetric) DDoS attacks, the attackers send a high amount of traffic, or request packets, to a targeted corporate network in an effort to overwhelm its bandwidth capabilities. This type of attacks does not exploit any vulnerability[8], it just floods the victim with a sheer quantity of traffic that victim can not flow, blocking completely the access to the end-resource(e.g., website, service). According to Arbor Networks [23], volumetric attacks are by far the most common type of DDoS attacks, dominating with 75.7%. The most common attacks for this category are:

  – **UDP Flood**

    UDP is a connectionless protocol that uses datagrams embedded in IP packets for communication without needing prior packet switching in order to setup communication channels between two hosts [48]. An UDP flood attack, abuses normal behavior at a high enough level to cause congestion for a targeted network. It consists of sending highly-spoofed UDP small packets at a high rate to random ports on the victim's system using a large range of sources IP's. This causes the host to repeatedly check for the application listening at that port, and when no application is found reply with an ICMP 'Destination Unreachable' packet consuming all of its bandwidth. This process exhausts host resources, which can ultimately lead to its inaccessibility [46].

  – **ICMP Flood**

    The ICMP, analogously to UDP is a connectionless protocol, based on the IP protocol and is used to diagnose network status (like when a datagram cannot reach its destination) [49]. In a ICMP Flood attack, attackers send highly-spoofed ICMP echo requests[9] at large enough volumes to flood a network [50]. Most devices on a network will, by default, attempt to respond with ICMP Echo Reply packets and the combination of traffic will saturate both outgoing and incoming bandwidth, resulting in a significant overall system slowdown.

  – Domain Name System (DNS) Amplification

    According to the giant of the Internet(i.e., Cloudflare) [51], in a reflection-based volumetric DDoS attack, attackers leverage the functionality of open DNS recursive servers in order to overwhelm a target or network with an amplified amount of traffic. It exploits vulnerabilities in DNS servers to turn initially small queries into much larger payloads i.e., a small amount of traffic can be used to generate gigabits of traffic, rendering the server and its surrounding infrastructure inaccessible. In this type of attack, traffic is generated indirectly, making use of open DNS servers and the IP spoofing feature causing the victim to be inundated with packets from these servers.

[8] Also known as non-vulnerability based attacks
[9] Also known as "pings"

As illustrated in Figure 2.6, the attackers identified by the IP A and B spoof target server IP address and send DNS requests to open DNS resolvers on the Internet. Each request is an UDP packet, often sent along with the "ANY" argument in order to receive the largest response possible. After receiving the requests, the DNS resolvers, which are trying to be helpful by responding, sends a large response to the spoofed IP address. These responses are reflected and amplified by a large factor as compared to the requests.

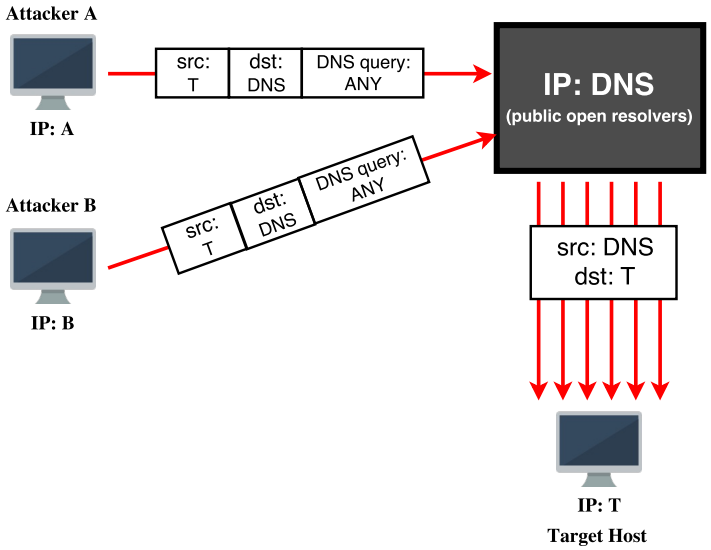



**Figure 2.6:** DNS-based volumetric reflection attack

For example, if a DNS server receives a 60 bytes EDNS[10] query, the response may be approximately 4350 bytes, resulting in an amplification factor of 73 [13]. It is easy to see that with the increase of the botnet's size, it is easy to generate several gigabits of traffic per second (Gbps), which ultimately will clogging the target network causing a denial-of-service.

– **UDP-based memcached**

Memcached is a system meant to cache data and reduce the strain caused by memory intensive services(i.e., disk or databases). Memchached can have both UDP and TCP listeners, with the particularity that these servers expose their UDP port (i.e., 11211) to external connections by default, meaning that any of these servers are not behind any firewall. Due to this and the fact that requires no authentication, when merged these conditions with the facility of falsifying UDP, it makes this service vulnerable to be used as a reflector. Furthermore, this vulnerability in the UDP protocol implementation of memcached servers, it also allows amplifying incoming packets with a factor of over 50,000, according to Cloudflare [52]. For example, as the enterprise cited, they were targeted for an attack of this type, where attackers sent 15 byte packets and memcached servers responded with 750000 byte packets in return. Recently, the world's first search

[10]EDNS stands for Extended DNS

engine for internet-connected devices(i.e. Shodan) [53] exposed to the internet more than 105,000 memcached servers, which can be abused for this type of DDoS attack.

- **Protocol Attacks**

  In protocol DDoS attacks[11], the attackers exploit weaknesses in the Layer 3 and 4 of the Open Systems Interconnection (OSI) conceptual model, present in Figure 2.7:

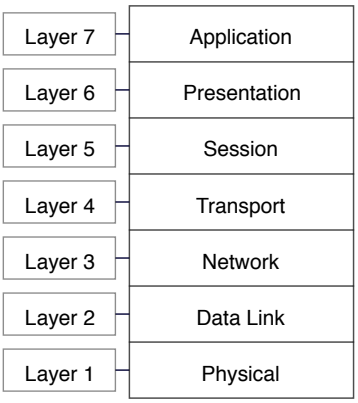



**Figure 2.7:** OSI Model

Protocol attacks consume all the processing capacity and memory of the attacked-target or intermediate critical resources like routers, firewalls, Intrusion Prevention System (IPS), Intrusion Detection System (IDS), load balancers, application servers causing service disruption [54]. Compared to the previous category differs in operating mode: instead of consuming the available bandwidth and congesting the network, it consumes the available resources. It also contrasts in how the target is crippled, i.e., the idea of a brute-force straightforward implementation is abandoned, where the target is flooded with several gigabytes of traffic, passing to an implementation that is based on the flaws that exist in some protocols allowing the consumption of resources available in the target host as well as the connection state tables. Finally, it differs by magnitude from bits per second to a higher level, packets per second(Pps). According to Arbor Networks [23], 11.8 % of the attacks belong to this category, where the most common examples of a protocol-based DDoS attack are:

  - **TCP SYN Flood**

    TCP SYN attacks exploit the inherent weakness of the three-way handshake involved in the TCP connection setup. Under normal conditions, as shown in Figure 2.8, there must be accomplished three distinct processes to establish a TCP connection:

[11] Also known as TCP State-Exhaustion Attacks

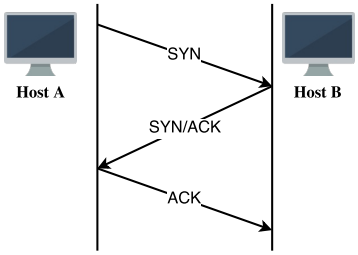



**Figure 2.8:** TCP three-way handshake

1. SYN - The application A sens a "SYN"(synchronize/start) packet to the application B in order to initiate the connection
2. SYN/ACK - The application B than responds to that initial packet wih a "SYN/ACK" packet, in order to acknowledge the communication
3. ACK - Finnaly, the application A returns an "ACK" packet to acknowledge the receipt of the packet from the application B.

After the three packet's sequence concluded successfully, the TCP connection is open and the application A and B are able to send and receive data. To create denial-of-service, an attacker exploits the fact that after an initial SYN packet has been received, the other application will responds back with one or more SYN/ACK packets regardless of the first SYN packet authenticity [55]. During the SYN flood attacks, represented in Figure 2.9, the attacker sends a high volume of SYN packets to the targeted application, often, depending on the botnet size, with spoofed source IP addresses. The targeted system, in turn, responds to each SYN packet and leaves an open port ready to receive the response in order to establish the TCP connection. While the target waits for the final ACK packet, which never arrives, the attacker sends continuously more SYN packets, causing the server to temporarily maintain a new open port connection for a certain length of time, considered as half-open, and once all the available ports have been utilized the target will be unable to proceed with its normal operation [56].

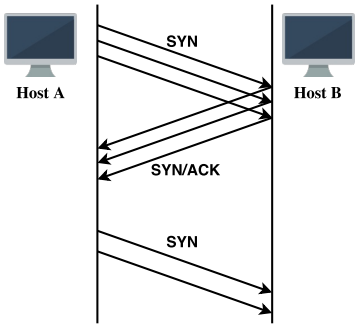



**Figure 2.9:** TCP SYN flood attack

According to Cisco [57], there are three variants of this attack vector: direct attack, spoofing attack and distributed direct attack. However, given the growth

of Internet of Things (IoT) devices[12], this kind of attacks are often carried out by a botnet, acting in a distributed way, making use of the inherent advantages and therefore increasing the complexity of mitigation.

– **TCP ACK Flood**

TCP ACK flood [13] is another protocol attack that misuse the TCP. In a TCP packet, the ACK flag acknowledges received data. However the attackers send continuously spoofed TCP packets, with the flag ACK set to true, at very high packet rates without the pre-establishment of a TCP session. Thus, there is impossible to correlate the received packet with any session within the firewall's state-table or server's connection list. By forcing state-tables to look up and trying to link these incoming packets to an existent TCP session, it depletes the victim's firewall and server resources.

– **TCP STOMP flood**

Simple (or Streaming) Text Oriented Message Protocol (STOMP) is a simple application layer, text-based protocol that allows clients communicate with others message brokers[14]. Similar to HTTP, STOMP works over TCP, allowing to create TCP sessions between two parties. TCP STOMP flood attack is a variation of the ACK flood attack since it is also based on the sending of highly-spoofed TCP packets with ACK flag set to true [58]. However it has two nuances: in addition to these packets also switches with packets with the PUSH flag set true, and, the respective packets are only sent if the pre-establishment of a TCP session exists. This requirement arose, since after some attacks were developed simple rules, such as block regular ACK floods, that are accomplished without completing the TCP connection process. Thus, attackers improved their attack vector, by implementing an evasion technique, that from obtaining a legitimate sequence number allows bypassing some of the network security solutions.

– **Ping of Death and Smurf attack**

Besides belonging to the same category and shares inherent properties, both attack vectors are classified as historic [59], [60]. Although they have played an important role in the past, nowadays are considered as solved vulnerabilities and so no longer prevails. A ping of death attack is characterized by sending packets larger than the maximum allowable size, causing the targeted machine to crash or freeze. While correctly-formed ping packets are very small(typically 56 or 64 bytes in size), IPv4[15] ping packets may be as large as 65,535 octets [61]. In the near past, systems were not designed to handle packets larger than this limit, making them vulnerable to packets above that size. Thus, when a packet is maliciously larger traverses one or more networks towards a target, the packet is fragmented into segments, each with a size smaller than the 65,535-byte limit. When the target

[12]Mostly routers and ip home cameras
[13]Also known as ACK-PUSH flood
[14]Program module that translate communication between different protocols
[15]Stands for IP version 4

system attempts to reassemble the fragments into a whole packet, the total size of its exceeds the size limit and may occur a buffer overflow which can freeze or crash the target system, preventing it from serving legitimate customers. Although this attack was mostly carried out through the ICMP protocol, using ICMP echo requests, there is no dependency between the ping of death and ICMP, and in fact, any protocol that sends IP datagrams can be exploited as well(e.g., TCP, UDP, Internetwork Packet Exchange (IPX)). Conversely, in smurf attacks there is a dependency on ICMP. However, it differs once it is an amplification attack vector that increases the damage according to the number of devices in the network, by exploiting characteristics of broadcast networks. Using a simple analogy to clarify how the attack works, it is possible to define a smurf attack as an employee who in bad faith due to problems with the Chief Executive Officer (CEO) of the company, sends an email to all his colleagues, posing as the CEO, asking explicitly to respond with the personal information to the respective email. In this way, every employee due to the obligation reflected by the company hierarchy, responds promptly with their personal information to the CEO's email, flooding him with unwanted emails. Transposing to use context, the attack is carried out with the packets falsification where the source IP address is set to the real IP of the targeted system. These packets are sent to an IP broadcast address, reaching every host device inside the broadcasting network, amplifying the attack through a scalar directly related to the number of networked devices on the network. Each of it receives a request and responds to the spoofed source address with an ICMP Echo Reply packet overwhelming the target system, resulting in denial-of-service to legitimate traffic.

  - **Generic Routing Encapsulation (GRE) Flood**

    According to Cisco [62], the largest network company in the world which developed the GRE, specified it as tunneling type protocol that allows the encapsulation of data packets, from a wide variety of network layer protocol, and routes them through a tunnel to a destination network over an IP network. On the right side of the virtual point-to-point, the packet is de-encapsulated by removing the GRE header previously added, and the payload is parsed. GRE flood [63] attacks are carried out by sending many GRE packets with a large amount of encapsulated random data which may lead to computational problems at the target system when it tries to de-encapsulate them and deal with the payload. This will never happen once they have no defined rules to handle these chunk of random data, exhausting the system resources [64].

- **Application-Layer Attacks**

  In Application-Layer attacks, the attackers exploit weaknesses in the layer 7 of the protocol stack. They target not only the well-known HTTP, but also HTTPS, DNS, Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) and other application protocols, exploiting their flaws. Typically, L7[16]attacks start by establishing

[16]Stands for layer 7

a legitimate connection with the target application server and then exhausts its resources (e.g., sockets, CPU, RAM, disk/database bandwidth). This process is executed through requests within this connection, making misuse of inner requests. That misuse can be concurrent connections, input parameters, session or injections of commands which trigger several back-end processes and transactions, such as database queries and Application Programming Interface (API) calls, that due to the resource consumption disparity between the client making a request and the server responding to the requests allows without much effort force a denial-of-service situation. Thus, attack vectors belonging to this category are the most sophisticated and stealthy [23] once they can be very effective with few machines generating traffic at a low rate sharing properties with flash crowd events[17]. It also differs from the other two categories since, with the rise in the OSI model, the attacks' magnitude becomes to be measured in requests per second (Rps). According to Arbor Networks [23], this attack category poses 12.4 % of the all DDoS attacks, which is mainly characterized by the following attack vectors:

- **HTTP Flood** The HTTP is the most widely used application protocol on the Internet [66], being, therefore, an attractive target for attackers. There are two varieties of HTTP flood attacks, based on the two main methods of it protocol:
  1. HTTP GET attack - in this form of attack, as the HTTP method indicates, multiple coordinated devices send numerous requests to the targeted server, which usually solicit large assets, that require more computational processing, such images, and files. When the target server is flooded with incoming requests and, considering that they have a legitimate HTTP payload, the server is unable to distinguish normal HTTP GET requests and malicious requests [67]. Thus, the server has to respond generically to all requests, triggering multiple underlying processes which exhaust server's resources, making it impossible to respond to additional requests from legitimate traffic sources.
  2. HTTP POST attack - antagonistically, this attack is characterized by sending multiple POST requests, which carry several data types such as images or form parameters, which, in order to be drained from the server-side, need to trigger complex processes such as pushing the data to a persistence layer, most often to a database. With the arrival of requests and the execution of the required database commands, the capacity of the target service becomes saturated and a denial-of-service occurs.
- **DNS Flood**

  Based on the same paradigm as other flooding attacks, a DNS flood attack target the DNS application protocol by sending a high volume of lookup requests to a particular domain's DNS servers, attempting to disrupt address resolution for that domain. Usually, these multiple requests involve querying for real records on the domain, mimicking legitimate traffic preventing any implementation of a priori rules to filter out malicious requests. In this way, DNS servers are easily

[17] Situation when a very large number of legitimate users simultaneously accesses a popular Website [65]

overwhelmed by the high volume of requests, and in the case of corporations that rely on others for the provision of DNS services, they are indirectly affected by the attack being compromised the availability of their resources, such as websites, API's or web applications, on a particular zone.

– **DNS Query Floods**

The operating methodology of a DNS query flood attack[18] can be explained using the snowball effect's analogy. It takes advantage, subtly, the resources of each agent machine, sending few requests, ultimately the snowballing number of queries will relentlessly torture and collapse the targeted server. Typically, these queries contain nonexistent randomly generated prefixed subdomain in order to pass straight through to the authoritative servers, which are, in fact, the target of DNS water torture attacks. These attacks, as shows Figure 2.10, exploit the improperly configured open-access recursive servers which when it receives a query for which it does not have cached data, like a randomized unique subdomain, relay the query to the appropriate authoritative server, acting as a transport layer, which makes malicious requests to reach their destination. Once the traffic that each bot generates is not intense, and each query appears normal, it makes difficult to filter spiteful from legitimate queries. Although these queries are small and, initially, insignificant, with the process advancement, the aggregate number of queries towards the targeted server becomes a tremendous flood, overloading the server by wasting resources looking up to nonexistent subdomains [68], until it becomes unresponsive.

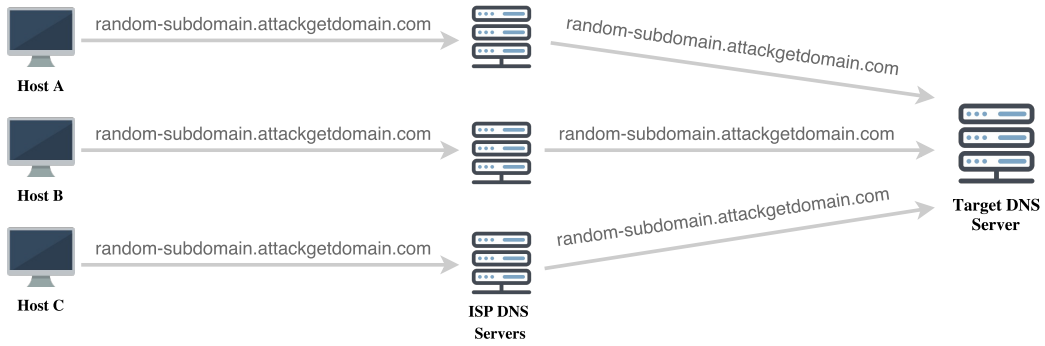



**Figure 2.10:** DNS query flood

According to Akamai [69], there is also a detail that could lead to a considerable disaster for many Internet users. Once the attack is based on intermediate local Internet Service Providers (ISP), to reach the targeted authoritative server, when it gets exhausted and the attack network keeps sending malicious queries, it leads to a domino effect on relaying recursive servers, which, repeatably, try to obtain a response from their upstream resolvers. However, this attack collateral effect leads to a major concern, since with the temporal advancement and non-existence

[18] Also known as Pseudo Random Subdomain, or water torture attack

of an authoritative server to respond, the intermediaries see their resources being gradually consumed until exhaustion, also leading to a denial-of-service situation for local users [70].

Despite the differences between categories such as the OSI operating layer, the focus on network or server resources, some attacks can be a combination of some categories taking advantage of the characteristics of each, that when aggregated provide an unpredictable multi-attack vector, where some of them are used as smoke-screen and the others carry out, effectively, the attack.

### 2.3.6 DDoS attack dynamics

Independently of attack categories, it's rates poses also a major threat that increases its output and take advantage of soft spots in mitigation solutions. There are several attack dynamics, which define its intrusive character that is related, not only with the agents number that constitute the attack platform, but also with attack frequency domain, including its activity and sleeping times. Typically, attack traffic flow-rate is inversely proportional to the number of bots. For example, given an offensive platform composed by five hundred agents, in order to generate five thousand packets it is necessary that each one produce ten. However, with a more extensive platform consisting of one thousand bot, each need only to produce 5 packets. Bellow, is presented a list containing some distinct attack dynamics [71], which precedes a groups Figures 2.11 that exemplifies each of dynamics, considering a max bandwidth of 500 KBps:

- Constant rate attack: it is a dynamics where there are no peak, the maximum rate is reach immediately and maintained until the master stops the attack. It is characterized by its highly intrusive nature.
- Increasing rate attack: similarly to the previous one, only differs in the maximum's reach rate, being later, following a linear growth.
- Pulse-wave attack: Also known as *hit-and-run* attack, is differentiated by its constant oscillation between the maximum rate and zero, resulting in square pulses. Typically, the uptime and inactivity is equal, making the attack periodic. Nevertheless the active and inactive periods can be parameterizable, allowing the master to control vector's intrusion degree.
- Gradual pulse-wave attack: as the name implies, the difference lies in the necessary time to reach maximum rate or zero, being gradual in both, but not necessarily the same.

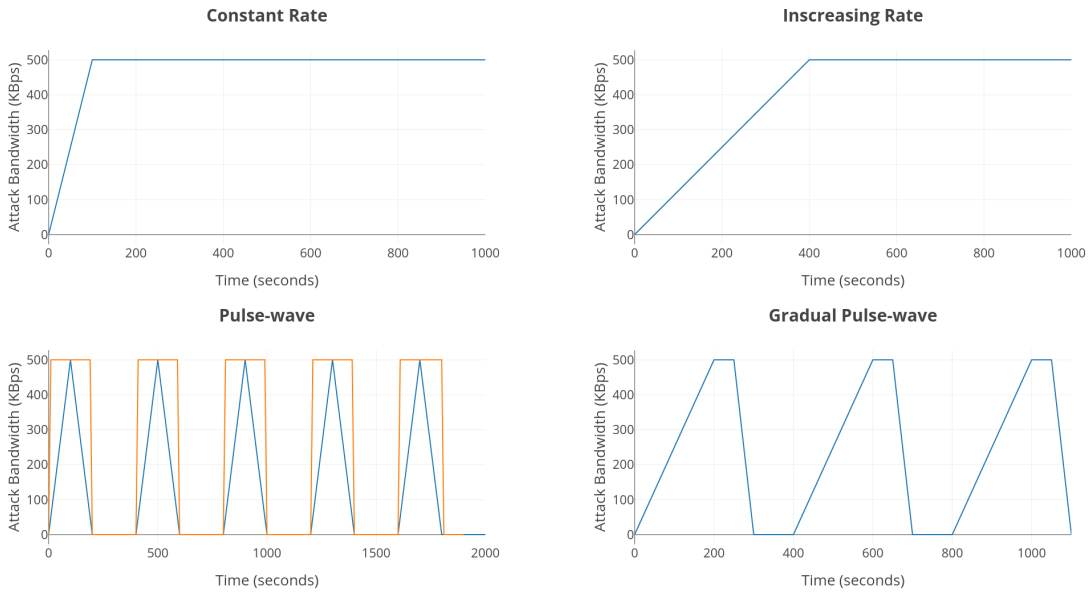



**Figure 2.11:** Representation of previously described attack common dynamics.

According to Imperva's white paper [72], *hit-and-run* attacks are the new emergence threat. This new attack sophistication allows to leverage botnet's resources, doubling its output and exploiting multi-targets. However, there is no consensus on what happens between peaks. Imperva analysts, affirm that during pulse-wave downtime period, the botnet doesn't shut down and used the interval between each pulse to mount a new attack on a different target. The downtime is maintained, but every pulse is directed to a different target. Instead, other experts, say that in the gaps, the botnet switches to another target directing similar short attack bursts, avoiding sleeping times [73]. Despite the divergent opinions of the other corporations, including Akamai and Arbor [74], several occurrences were recorded during the second quarter of 2017 that used this technique regularly. Besides the resources portioning advantage to launch multi-target attacks, Imperva proves that this kind of initiatives can cause problems and ultimately disrupt hybrid mitigation solutions. These are distinguished by its mixture between on-premise hardware and off-premises cloud-based scrubbing. The on-location hardware is the first line of defense. It is configured to trigger an alarm, when faces an attack exceeding its capacity limits, which activates the cloud-based DDoS protection. Its main objective is to scrub incoming traffic applying diversion processes. However, this step never initiate, once pulse-wave attack hit fastest and furiously the first barrier, congesting the network pipe, cutting off the communication between on and off-premise components. Even if the cloud is configured to activate automatically, it still has to analyses the traffic from scratch, without any indication about its signature due to the lack of communication, before filtering it.

Despite the different dynamics, their frequency of use and intrusion degrees, which are high result of maximum rate's allowed by the devices's network card, it is possible to reverse this suspicious posture, making it more intelligent and similar to a human behavior, by reducing the coordinate axis. Instead of using the maximum rate, should be limited by packet's number,

or ideally the bot must learn from device's behavior on the network, including external communications. Based on the normal behavior, the bot should be able to include malicious traffic during connections out of the network, making it imperceptible tk differentiate traffics. Nowadays, this approach was not achieved, and its not necessary once there are plenty of devices across all the Internet which are not properly patched and maintained, being easily subverted, avoiding the need to implement more complex solutions and concern with the intrusion degree and exposure. The attack with the dynamics that most equals the ideal is designated: *low-rate* DDoS attacks. These are identified by its long and quietly operation mode. The main purpose of these attacks is to degrade the victim's resources gradually, by creating a couple of connections over a period of time, leaving them open on the target for as long as possible. For example, *Slowloris* is an attack tool that is available on the Internet [75], which follows this approach to disrupt web servers, with minimal bandwidth. After opening the connections, the tool keep sending small packets or keep alives in order to prevent session from going to idle timeout. Often, these attack vector are intermingled with volume-based attacks, that due to its stealth character are not only tough to detect but also to block.

## 2.4 A glance to the past of DoS and DDoS attacks

For the past 25 years, the DDoS attacks have received increasing attention from both network operators and general media while the basic vulnerabilities that make attacks possible have been recognized. Computer Emergency Response Team Coordination Center (CERT/CC) published the first advisory regarding a UDP port DoS attack (CERT CA-1996-01 [76]), in February 1996, a beginning of a year that would be a landmark in the history of DoS attacks. Seven months later, New York City's original ISP, Panix, was hit by a SYN flood denial-of-service attack that took it offline for several days. The attack started on September 19 and immediately CERT published another advisory alerting of a TCP SYN flooding DoS attack using spoofed source IP addresses (CERT CA-1996-21 [76]). It would not take 3 months to publish the last DoS attack advisory of the year (CERT CA-1996-26 [76]). This time the advisory dealt oversized ICMP echo request packets that when received by some operational systems could lead them to crash, freeze, or reboot, resulting in denial-of-service. This particular attack is often referred as the Ping of Death. The year 1998 began with two major publications differentiated by its scope, the first one related to the discovery of a new vulnerability and another related to defense measures that defeated the current attack vectors. On January, CERT issued another advisory, detailing a so-called Smurf DoS attack (CERT CA-1998-01 [77]). It is important to note that, although the attack is launched from a single machine, as explained in Section 2.3.5, it might result in multiple nodes sending traffic to the target and therefore should not be categorized as a DDoS attack. Days later it was released the RFC [78] which detailed how network administrators could defeat DDoS attacks via anti-spoofing measures.

This would eventually become the best current practice adopted by many networking vendors. A year later, on August 17, 1999, it marked the year with the first large-scale DDoS

attack when a DDoS tool called "Trinoo" was deployed in at least 227 systems, according to Bhuyan et al. [15], to flood a single system at the University of Minnesota, swamping the target network and rendering it unusable for over two days [79]. However, despite this important reference, it would quickly be forgotten due to a sequence of attacks that occurred in the beginning of the year 2000, and surprisingly disrupted the services of large corporations, something never seen in the history of the Internet. As reported by BBC News [80], in February 2000 a denial-of-service attack took down "Yahoo!" for approximately three hours. However, just one day later, websites like eBay, Buy.com, CNN.com, Amazon.com, and other eCommerce[19] sites, suffered also a heavy attack which was similar to the previous, either in duration either in magnitude. For instance, in some cases, the overall amount of incoming data exceeded 1 gigabit per second [81], a magnitude completely unexpected compared to current server processing capacity. Thus, it was quickly inferred that in order to obtain an attack of this size, multiple computers would have to be used, making this the first major DDoS attack, that had a financial overall damage estimated at 1.7 billion dollars [82]. From this incident, denial-of-service took on new visibility and importance, once if the company who had probably the greatest web resources could be taken down then anything can be brought down. After this milestone, many devastating DDoS attacks have kept occurring on the Internet, although its attack vectors along with the defense mechanisms haven't evolved much.

In October 2002, the thirteen DNS root servers operated by U.S. government agencies, universities, nonprofit organizations, and companies were simultaneously under attack for about an hour, of which nine of them become unavailable for regular Internet traffic [83]. The remaining root servers supported the attack and ensured that the holes generated by the loss of more than half of the servers would not significantly affect Internet performance. Which was successfully accomplished, allowing end users to continue with their normal operations without noticing any change, despite having been the most serious attack ever attempted on a key piece of the internet structure. At this time, there was evidence that DDoS attacks were an emerging threat that had the ability to overthrow both the servers of the large corporations as well as the infrastructure of the internet. Subsequently, since 2004, attacks motivation has been shifted to the sensitive field of economic crime harming the victims' reputation in the eyes of the public, which are gaining awareness of the actual security, and due to this, incidents of denial-of-service began to be overshadowed by the mainstream media [81]. In addition to the economic motivations, a trend also started towards gaming companies. In February 2007, more than 10.000 online games servers belonging to games like "Halo" and "Counter-Strike" were attacked by "RUS" hacker group, which took advantage of more than a thousand of computers located in the republics of the former Soviet Union [84]. Furthermore, as stated in [85] , this year would also be noticeable by the occurrence of attacks of a political nature, due to a series of diplomatic tensions between the former Soviet Republic of Estonia and Russia. The cyber attacks escalated after the Estonian government decided to relocate a Soviet World War II memorial statue, leading to a start of a spree of denial-of-service attacks against

[19] Stands for electronic commerce

the Estonian organization's websites, including the parliament, banks, ministries, newspapers, and broadcasters. The year 2008 would also go down in history with the beginning of a legacy that still prevails today, with the appearance of a group characterized mainly by its ethics, aiming at freedom of expression, human rights and freedom of information. This hacktivist group is widely known as, *Anonymous*.

The attack known as "Project Chanology", was it's first appearance, which knocked the official Scientology website[20] offline. It was a response to the church's attempt to remove a highly-publicized Tom Cruise interview video from the Internet [86]. It would take 2 years for a new appearance, this time with a new campaign titled as "Operation Avenge Assange"[21] once again characterized by the claim of the group's ideals, namely freedom of information and government transparency. The operation came as support to Wikileaks[22] since it was under intense pressure to stop publishing secret United States diplomatic documents. According to [87], many companies around the world have adopted anti-wikileaks behavior, prompting the group to take retaliatory action against these companies. For instance, many financial organizations, such as MasterCard, Visa, and Paypal, which blocked payments to Wikileaks were one target of these attacks, along with EveryDNS.com due the deleting of the WikiLeaks DNS record and so on.

The infamous group's journey continued with a new operation against Sony's websites, announcing it's intent to defend the hacker who jailbreaks[23] the PlayStation 3, and got sued days later by the company which took offensive actions against free speech and internet freedom by gaining access to the IP addresses of all the people who visited the hacker blog where was published a detailed instructions of the respective jailbreak, according to [88], [89]. However, days later, another group of hackers, break into PlayStation Network (PSN) and Sony Online Entertainment exfiltrating millions of customers records. Immediately, Sony camouflaged the huge data breach by shutting down the PSN claiming that it was suffering from technical issues. Nevertheless, it was a time-bound measure given the sheer size of the data exposed, and the length of time the PSN was unavailable, Sony executives were forced to admit [90], apologize and ask for help from the FBI[24] and external security companies.

At the beginning of 2012, another operation appeared that would also be for the story. Once again executed by the Anonymous group, this time against the shutdown of file-hosting service "Megaupload" by the U.S. Department of Justice which indicted the executives of numerous illegalities, including criminal copyright infringement and conspiracy to commit money laundering. The "Megaupload" operation was set up, which included about 6,000 people who supported the group's punitive actions by using LOIC[25] against the company responsible for the "Megaupload" case as well as 10 other high-level government websites [91],

[20] https://www.scientology.org/

[21] Also known as Operation Payback

[22] Organization that publish secret information, leaks, and classified media provided by anonymous whistle-blowers

[23] Privilege escalation for the process of removing software restrictions

[24] Stands for Federal Bureau of Investigation

[25] An open-source software tool that aims a massive flood a targeted site through junk TCP, UDP and HTTP GET requests

such as US Department of Justice, US Copyright Office, and FBI.

The year 2013 arrived and with it 2 meaningful events, chronologically separated. The first came in March and would be a turning point as the attacks escalated to a completely unexpected level never seen before. Spamhaus, an international non-profit dedicated to battling spam, saw his servers being flooded with packets at a 300 Gbps flow rate, as the target of a DNS amplification attack. Thus began a trend of growing attacks in both magnitude and number of occurrences.

The second important moment occurred three months later with the continuation of an operation that had been created in 2012. This operation was a set of attacks targeting various American financial institutions (e.g.,Wells Fargo, U.S. Bancorp, Bank of America) operated by by a group calling itself the "Cyber Fighters of Izz Ad-Din al-Qassam". The operation consisted of three phases, based on all of them, an ideological/religious motivation. The attacks were carried out using a web version of LOIC, designated as JS LOIC[26], that was also used on operation "Avenge Assange", allowing volunteers to also be an active part in the attack [92]. Subsequently, in 2014, has been created a black hat hacking group named "Lizard Squad" which started disrupting online gaming services since the creation's year with the purpose of raising internet security awareness and for simple amusement. One of the most charismatic was the attack on the PlayStation Network and Xbox Live on Christmas day, with the particularity, that the attacks had been announced weeks earlier. The attack was executed with the aid of a tool created by the group itself, called the Lizard Squad's Lizard Stresser[27]. This tool is a DDoS botnet [93], where each infected machine tries to propagate via telnet brute forcing by trying to connect to random IP's and then attempt to log in via telnet using a list of hard-coded usernames and passwords. Successful logins are reported back to a pre-defined command and control server, for later assimilation into the botnet, and subsequently to receive commands from the master. In addition, it also has attack vectors that aim to deplete the resources of the network (e.g., UDP flood, DNS amplification), and the resources of the servers (e.g., TCP SYN flood). Besides the creation, the group also conducted a marketing campaign for promoting their DDoS tool, allowing its use by third-party users. Beyond Lizard squad attacks, a new group of computer criminals arises entitled as "DD4BC"[28]and threatened targets, mainly financial institutions as stated by CloudBric [94], with massive DDoS attacks unless they pay a ransom using bitcoins. Each campaign launched by this group begins with an email that informs the victim of a low-level DDoS attack currently underway against the victim's website, and then, demanded a ransom paid in Bitcoins in return for abstaining from launching a larger DDoS attack, tipically, a DNS amplification or a TCP SYN flood. It was about to arrive in 2016, the year that would be known as "The year of DDoS attacks". The most scourged year ever, with a dramatic increase in peak attack size and frequencies. Of all the events, it is possible to highlight five of the most important, ordered chronologically:

[26]Javascript version of LOIC
[27]Or simply Lizard Stresser
[28]Stands for DDoS for Bitcoin

- The US candidate's campaign websites

  On April 1, once again, the hacktivist collective sought to take down the Donald Trump's websites, relating to its hotel chain and presidential campaign, as well as its email servers, trying to defuse Trump's candidacy to the White House and damage his brand. Not having effect, around the time of the Election Day, attackers made another move by again targeting the political candidates. This time they leveraged a Mirai IoT botnet to flood the campaign websites for both Hillary Clinton and Donald Trump, mostly with HTTP requests.
- Rio Olympics

  Three months later a campaign was launched against organizations affiliated with the Rio Olympics, making use of the tool previously made public (i.e., Lizard Streeser). Throughout the operation, in addition to this tool was also used several IoT botnets that added firepower to Lizard Streeser, including requests targeting the IP protocol Generic Routing Encapsulation. The attack ultimately peaked at 540 Gbps. It is also important to notice that, given the magnitude and longevity, it was expected that the logistic and media coverage would be disrupted, however, due to the mitigation measures provided by Arbor, was possible to International Olympics Committee keep all the systems up and running.
- Brian Krebs

  On September, the blog of information security investigative reporter Brian Krebs[29] experienced a DDoS attack. According to its post [63], the attack was unusually powerful, placing the peak attack traffic at around 620 Gbps. In addition, Akamai, the company that protects the website from such digital sieges, reported that the attack method didn't rely on amplification techniques, and instead followed the same method used in the last instance during the campaign against Rio Olympics, by leveraging GRE traffic along with SYN and HTTP floods. This was the second campaign based on Mirai botnet that would mark the history of denial-of-service attacks in several indicators.
- The French cloud computing company OVH

  It would not take a day for the Mirai botnet to further affirm its position, when 152,000 infected IoT devices, launch the largest DDoS attack campaign ever reported. In this way, OVH, the third largest internet hosting company in the world, was hit with two massive simultaneous DDoS attacks peaking, at least, 1.1 Tbps [95]. A few days later, on 21 September, the source code of the Mirai botnet goes public, signing its contribute on the denial-of-service history.
- DYN

  Precisely, a month later, Dyn, a US-based DNS provider that serves many companies (e.g., Twitter, Netflix, Reddit, CNN, Spotify PayPal, Pinterest), was attacked by the Mirai Botnet which despite seeing its code available, had never been carried out an attack using this vector, the "water torture" attack. According to the company's summary [96], it is estimated that the attack involved 100,000 malicious endpoints, majority IoT devices, leading to an increase in the magnitude of the attacks, to 1.2 Tbps.

[29] Available at https://krebsonsecurity.com/

A year later, corroborated by the Arbor study [23], it is possible to conclude that 2017 was dominated by three trends: politically motivated attacks, attempts to cash in on the soaring price of Bitcoin and attacks targeting the entertainment and gaming sector. The last two trends are directly related in that they share the same objective: to extort money. This approach was dubbed "RDoS"[30] and is based on the paradigm, "demonstrate force, demand a ransom", previously introduced by "DD4BC". However, it gain notoriety in 2017, with several groups making use of it by threatening companies which would suffer substantial losses if their resources are unavailable such as banks, Initial Coin Offering (ICO) platforms, etc. For instance, in June, Armada Collective, a similar group to DD4BC, demanded about $315,000 from seven South Korean banks in exchange for not disrupting their online services. The pattern would be used again, two months later, by targeting the Americas Cardroom online poker site. They refused to pay a ransom and as results of the attack were forced to delay a poker championship that was already underway. In addition to the gaming and entertainment companies, ICO platforms also suffered due to the commercial popularity of bitcoin and everything associated with it. The broad availability of these platform guarantees reliable and secure transactions, while DDoS attacks are aimed at breaking the operability of the service and thus discrediting it. For instance, Bitfinex, the world's largest US dollar-based Bitcoin exchange, was a recurrent target, seeing it services paralyzed often, allowing attackers to profit from Bitcoin price fluctuations due to denial-of-service. The two major attacks occurred in June and October, when the platform launched two new cryptocurrencies, IOTA [97] and Bitcoin Gold (BTG) [98] respectively.

Finally, in relation to politically motivated attacks, the main onslaught was pointed out against the Spanish government, due to the issue of Catalonia. The Anonymous launched an "Operation Free Catalonia" campaign which took down the website of Spain's Constitutional Court through 3 attack vectors: TCP SYN floods, UDP floods and HTTP floods, as stated in [99]. Besides that, Ministry of Public Works and Transport's website was defaced to protest injured people in Catalunia, as Anonymous explicitly put in the content of the site. There were others such as the attack during the parliamentary elections of the Czech Republic, but the consequences were minimal, being cataloged as nuisances.

Most recently, in late February, Github suffered the second-largest attack in history, being bombed with 1.35 Tbps due to a memcached-based attack. According to it incident report [100], their services were unavailable during 10 minutes, an outstanding short time, completely unexpected taking into account the magnitude of the attack, which not caused worst problems thanks to the detection and mitigation mechanisms of Github and its partnership with Akamai. Two days later, Akamai itself reported [101] that several customers were being targeted for similar attacks with the nuance that this time was associated with another purpose, such as attack payloads requiring 50 XMR (Monero coins), approximately $15,000. Once again, two days later, on March 4, 2018, the same attack methodology would again be used, this time aiming to an unnamed US service provider. Arbor Networks, the company responsible for this service provider's protection, confirmed a 1.7 Tbps [102] reflection/amplification attack

[30]Stands for Ransom DDoS

beating the previously established record, making it the biggest attack ever, at least to this day.

In short, it is possible to recognize an evolution, over the last 25 years, in DDoS attacks. Along with technology in general, also nature of the attacks evolved, having nowadays a clear tendency towards the black side, allowing to obtain profit from incidents of denial-of-service, or from the threatened to suffer one. At the same time, it is possible to verify a sophistication in the methodologies of attack, aiming always to innovate in relation to the old ones and to the present mechanisms of defense and protection. To make matters worse, in addition to attack's nature and the devastating attacks' vectors, botnets have also evolved immensely due to the lax password management practices and security vulnerabilities found in IoT devices geographically distributed across the globe. With great motivations, unexpected attack vectors and a large number of participants, great calamities arise, and there is a direct relation between this association of conditions and the attack's size. According to the story it is possible to conclude that there is a clear pattern, which follows an increasingly growing function of raising the peak of traffic recorded between attacks, being now at 1.7 Tbps. Due to this gradual growth, both in frequency and magnitude, DDoS attacks are no longer a nuisance but a major threat against the availability of websites, online services and applications. Currently represent one of the biggest cyber attacks [103], with proven evidences and due to the convergence of its nature into the sensitive field of economic crime, the DoS incidents were deliberately not widely publicized as they call into question the reputation of the victim in the eyes of the public who is increasingly aware of security in general [104], [105].

## 2.5 DDoS Defense Challenges

Designing proper defense systems against DDoS attacks is not a trivial problem for network administrators and network security developers. Due to the constant sophistication of the DDoS attacks, cyber criminals are usually, one step ahead of the current defense mechanisms, which are not able to handle all types of network attack vectors in near real time. Since all DDoS attacks are based on a large number of compromised nodes, which are intended to attack network, system, or service resources, it is imperative to detect in advance any preparatory activity for an attack by mitigating it immediately. It is also necessary to maintain a separation so that the mechanism does not suffer with the high collateral damages and in the last instance, that is not the target of an attack. From a high level perspective the designing process of any DDoS defense system fight against technical and social challenges, according to [31]. From the technical point of view, the main challenge arise due to the distributed nature of the DDoS attacks, that by using several geographically dispersed action points, both for agent selection and target definition, and once the internet is administered in a distributed way, the deployment of a defense solution covering all stages of the operation of an attack can not always be carried out and guaranteed. Thus, it is unrealistic to assume that a single point could reach the total defense goal, once several networks have different requirements analyzes counting on their policies, preferences and budgets, making it impossible to establish

an ideal solution throughout the Internet. Hence, according to Mirkovic et al. [106], there must be multiple defense systems, deployed in various networks that must have the ability to constructively relate to grow and negotiate different policies. It is fundamental that each solution is not designed as complete, on the contrary, it should be flexible as a set of distinct units with an associated business value that together constitute the solution. In addition, in this way, it promotes interoperability and continuous evolution and deployment leading to social challenges. The DDoS defense systems should follow some patterns of deployment to be effective, by working under partial deployment but also must be incrementally and continuously deployable. Along with this, the wide deployment on networks that share the same needs, must be guaranteed at a large scale, contributing to the system scalability whereas it must also provide means for participants(e.g., hosts, routers, or networks in the Internet) to dynamically join and leave the system without endangering the system's operability. However, in spite of the different principles there are some factors that determine the probability of a wide deployment. Assuming the needs of a customer, the defense system have a good economic model, providing a strong economic deployment incentive to each customer which must have direct monetary benefits when using the system or at least reduce the risk of economic losses. In addition, as the degree of implantation increases, the client experiences an increase in benefits. Last, but not least, each defense system must meet the needs of each organization/customer, however it should be clear that it is not a strict requirement, since any product that improves the current state is enough.

## 2.6 Taxonomy of DDoS Defense Mechanisms

The defense systems that can be categorized according to a set of factors. Following the DDoS classification of the defense mechanisms, reported by Bhattacharyya et al. [36], there is a defense's segmentation according to each attack features, that might be combined together to effectively and completely solve a specific problem. Given the various phases of the generic mode of operation of a DDoS attack, there are several approaches to confront these cyber attacks.

### 2.6.1 Based on Approach

Based on the activity level, DDoS systems can be labeled into four types:

- **DDoS detection**

  A DDoS detection solution is characterized by constantly monitoring the network or system that detects unusual or malicious activities along with policy violations, and which notifies the network administrator using email, paging or logging the occurrence. It is a set of techniques to identify suspicious activity at the network or host level. For instance, an Intrusion Detection System (IDS) like *Suricata* is a solution that fit with this approach.
- **DDoS prevention**

A DDoS prevention system is nothing more than an improved version of detection systems in that, in addition to monitoring network or system traffic and identifying malicious activity, it also enables the blocking of detected intrusions. Intrusion Prevention Systems (IPSs), like *Snort* or *Check Point IPS*, typically contain a set of actions that do not exist in IDSs like discarding malicious packages, reset the connection in addition to dynamically blocking traffic from offensive IP addresses by leveraging firewall capabilities, which is also a defense system used in enterprises network architectures. It is a solution that establishes a barrier between two networks, one that is normally secure, controlled and trusted, and another external that is untrusted, for example, a company network and the internet, respectively. Firewall solutions[31] can monitor all the network traffic and confront with defined policies and rules, and any attempt to bypass security regulations activates alert mechanisms and adding rules to deny traffic. This actions are aided of simple packet-filtering techniques such as Shallow Packet Inspection (SPI), which is a type of data processing in which packet headers are parsed, operating up to the layer 4, inclusive, of the OSI model. Usually, firewalls apply the Packet-Based per Flow State (PBFS) method [107] by analyzing the first four layers creating a 5-tuple containing the source address, destination address, source port, destination port and the transport protocol. On the other hand, for more extensive inspections and analyzes there is other data processing type designated as Deep Packet Inspection (DPI). Which is a meticulous inspection up to Layer 7 of the OSI stack, being able to examine both the headers and the payload, without any restriction. It is commonly used in organizations with refined firewalls that have IDS/IPS components, such as *Cisco PIX firewall* [107], and at the ISPs at level of IPSs. However, often this use by network operators generates discomfort on the part of customers and employees related to privacy issues and legal questions [108]. According to the directive 2002/58/EC [109], European citizens have the right to privacy of communications, however, most of the times, the ISPs instead of providing the sites visited to the partners of the advertisement, sell the users' behavior on the Internet when collecting and aggregating their web browsing activities raising serious privacy concerns [110].

- **DDoS response**

  A DDoS response system is an extension of detection solutions, as in addition to sharing continuous monitoring processes of system health, each alert generated by the detection system for the presence of malicious or unauthorized activities triggers a process to react with countermeasures returning the system to healthy mode. There is a direct dependence between the reaction capacity and the detection system, preventing a reaction to be taken without the problem being noticed and understood. The faster the attack is detected, the faster the countermeasures will be determined to deal with the particular attack, reducing the effects of the attack while increasing the effectiveness of the reactive measures [31].

- **DDoS tolerance**

[31] Can be hardware, software or both

A DDoS tolerance system takes a fault-tolerant posture to defend against DDoS attacks. There should be mechanisms that prevent intrusions and attacks from leading to complete disruption of the system, so that the system functions in a reduced manner, but at a reasonable level.

### 2.6.2 Based on Defense Infrastructure

Defense systems can be developed based on the infrastructure, which can be host-based and network-based [36].

- **Host-Based DDoS Defense**

  The focus of this type of defense is the individual host, meaning that each of the hosts on the network has housed a defense system. Thus, a host-based DDoS Detection and Prevention System processes the data underlying the monitoring and all activities on the computer, such as event and kernel logs. Typically, these systems can also control computer's resources in addition to continuously oversee its state ensuring that all actions follow a normal pattern and otherwise flag anomalous use.

- **Network-Based DDoS Defense**

  A network-based DDoS defense system operate at a higher-level, examining data exchanged among computers. These systems are characterized by their ability to monitor and audit in real-time, as well as on a scheduled basis, distributing CPU utilization by providing a flexible means of security administration. To assist the monitoring process, several sensors are distributed across the network by sniffing the packets, allowing a complete map of the packet crossing in the network. The defense system compares packets with expected signatures to encounter unusual or abnormal behaviors, which depending on the system approach, described in Section 2.6.1, can lead to various responses and actions, from alerts to the network administrator to appropriate countermeasures.

### 2.6.3 Based on Defense Location

The DDoS threat can be countered at different locations among the networks. The defense mechanisms can be deployed in 3 different places [9]: victim end, intermediate-network, and source end. Each possible location has its own advantages and disadvantages.

- **Victim-End Defense**

  Generally, most DDoS defense systems are deployed at the end of the victim, on the routers of its network(i.e., networks providing critical Web service) [15]. Since it is the convergence's point of traffic where the largest damage occurs and therefore, what motivates the investment in a defense system. A victim DDoS defense system significantly facilitates the detection of attacks, as it can closely study the victim, define his behavior and report any anomaly. Due to the high arrival rate of packets and the consumption of resources, the anomalies stand out abruptly through unexpected peaks of traffic. However, the response's spectrum is confined. Once the defense system is in the path of full force attack, the target point can be quickly moved from the predefined

target to the defense mechanism. Another disadvantage of systems deployed in the victim's network is the limited amount of processing and storage that defense system has available as consequence as the large volume of traffic. At this point the differentiation of legitimate flows and attack is complex not only by the strong aggregation but also by the unavailability of computational resources that are necessary to make a sophisticated traffic profile, extracting statistics from each of the flows [10].

- **Intermediate-Network Defense**

  As the distance to the target increases, it is possible to identify two relationships with different proportions. The first is directly proportional to the bandwidth consumption, which is expected, since by moving the defense upstream, increasing the distance to the congestion point, the bandwidth as well as the resources available will increase. On the other hand, there is a trade-off between the distance to the attack's target and the accuracy of detection of this. Often these intermediate network defense systems are installed on the core routers and they are autonomous and independent in regard to attack detection, since they neglect any collaboration with the surrounding routers, to focus on themselves and only detect anomalies in a self-sufficient manner. However, since these are the main Internet routers, they manipulate large volumes of traffic, highly aggregated, causing the detection process to be difficult, and it is only possible to detect large-scale attacks. On the other hand, if detected, the responsiveness to suppress is high thanks to the abundant network resources. Nonetheless, it is necessary to keep in mind that only traffic rate limitation is possible since the creation of traffic profiles is computationally time consuming and these routers can not waste memory or processor cycles for all the streams that pass through them. Despite these disadvantages, the main difficulty of using a DDoS attack defense system in core routers is their deployment. In order to maximize the detection accuracy of all attacks it is necessary for all Internet routers to employ systems using this approach, however it is obvious that the practical implementation of this phase is unattainable [36] because it requires the reconfiguration of all core routers and many of them may not support the most current mechanisms.

- **Source-End Defense**

  Source-end defense systems are deployed at the attack's sources to prevent network users from generating DDoS attacks. Following the proportionality relationships previously presented, the main difficulty with this approach is that, detecting DDoS attacks at source end is not trivial [31], once sources are widely distributed and this devices behaves almost similarly as in normal traffic . On the other hand, if the attack is promptly detected, it is the point where the effectiveness of response and the availability of resources is greater. Also due to this availability, systems deployed at the source end are capable of using complex processes for profiling, facilitating detection and in turn minimizing collateral damage [15].

### 2.6.4 DDoS Defense Goals

Apart from the architecture and strategy of the defense mechanisms, there are basic objectives that are transversal to all DDoS defense systems and that must be achieved. The primary goal of DDoS defense is to provide good service to legitimate clients in the presence of a DDoS attack. Ideally, clients should not perceive any degradation in quality of service while the attack is in progress. The defense mechanisms must effect a separation of flows, avoiding that legitimate packets are dropped causing collateral damages. Since attackers often use obfuscation techniques, while mimicking normal traffic, it is normal for legitimate traffic to resemble attack traffic, as can be seen during flash events[32] [111], thus making it difficult to deal with collateral damage. The second objective is to alleviate the outcome of the attack so that resources can be properly channeled to serve legitimate clients. In order to achieve this preservation, it is necessary that the defense system be comprehensive in relation to the ability to handle various attack vectors, and that it is effective in the ability to respond. Thus, completeness is mandatory in detection but also in response [31]. If the detection module does not recognize a pattern of attack it is presumed that it can not detect it and perhaps the response will not be adequate or even non-existent and the attack will succeed. This inability to detect makes the system ineffective in the same way that failure at the response level allows the victim to suffer seriously from the attack, something to be avoided when designing such a mechanism. Another aspect that should be avoided in conception a defense system is the system's resistance to attempts to bypass it or deactivate it. The system must therefore be reliable and high security. Other imperative objective is the ease of deployment and respective operating expenses[33]. There must be a direct proportionality between the costs associated with defense systems and the direct benefits of their use. When it comes time to buy a commercial solution it is obvious that there is an initial capital expense[34] related to the hardware and software needed. Subsequently, administration costs and ongoing support[35] arrive, which depending on the approach of the defense mechanism may be more frequent or not. Finally, from a lower level perspective, it is important that the false-positive rate is low. According to [112], a false-positive occurs when the detection part of the defense system reports a network event as a serious problem when it actually is not a problem. Thus, DDoS defense mechanisms should target only true DDoS attacks. Otherwise, it may cause collateral damage to victim network and legitimates clients, or even, in the case of being regular, may indicate an incorrect pattern in the network administrator that when confirming that the alerts are false, starts ignoring them or turning off the system [31].

[32] Drift from flash crowd, and refers to a condition when the rapid disclosure of news about an event leads to simultaneous mass access, overloading the respective website servers

[33] Commonly known as opex

[34] Also Known as Capex

[35] patches for security enhancements, performance or even characterization of new attacks.

## 2.7 Source-End Defense

Ideally, DDoS attacks should be interrupted as close to the source as possible. This dissertation is based on this resolution allowing to leverage the multiple advantages [113] over intermediate network and victim-end defense approaches, as described in Section 2.6.3. By assuming a defensive position close to the attacker and limiting the attack flows it generates, it prevents the attack from proceeding and that the Internet as a highway from the target network attack is overloaded with large volumes of traffic, avoiding the general congestion in addition to increasing the resources available to legitimate users and services.

Another advantage deriving from the use of a defense system at source is the reduction of collateral damages. The majority of DDoS defense systems respond to the attack with traffic policing, using approaches such as filtering or rate limiting [31]. However, legitimate traffic suffers collateral damages induced by these measures. Instead, if the filtering acts at the source level, even partial blocking allows traffic from uncompromised networks to proceed, not suffering any change due to possible impediments in the victim.

Another aspect that notably motivates the development of DDoS defense systems in source is the possibility of implementing more complex and modern defense strategies. In an attack scenario, the further away from the target, the less the traffic volume to which the routers are subjected, and therefore, there is an abundance of computational resources at the source that allows the use of more sophisticated detection mechanisms. In addition, many DDoS defense systems have in their response module, traceback procedures that aid in terminal identification and rule enforcement. At the source, for example in a corporation's network, it is the closest point and therefore facilitates traceback and investigation processes in order to alert the network administrator that one of the devices is compromised and is participating in an attack. Thus, the decision-making and enforcement process is accelerated, preventing participation and hence possible damage to reputation and, in many cases, legal proceedings, including the inherent monetary costs. Despite the debate over the legality of DDoS attacks as a criminal deed or act of civil disobedience, each country has its jurisdiction [114]–[117] in accordance with its view on DDoS incidents. For example, in United Kingdom each person who is an active part, whether it is intentional or not, of a DDoS attack is at risk of being charged with legal offenses at the legal system.

Corporations and universities are often exposed to association with activities of this nature. In order to avoid affiliation and possible legal complications, many of them, not only in UK but also in the USA, have been implementing a computer use policy [118], [119]. In this way, any malicious activity performed by an institution computer is the current user's responsibility. But, should intent or effect be prioritized? It does not matter, since regardless of whether the employee/student was persuaded through social engineering to perform malicious actions within the organization, or if the network administrator forgot to patch a vulnerability and the company negligently cooperated in an attack, a source-end defense system should cover all cases, preserving reputation and revenue.

On the other hand, source-end defense also faces some difficulties, in terms of detection

and response selectiveness. Attack detection at the source-end has an elevated degree of difficulty due to the highly distributed nature of the attacks as it allows only to observe a small portion of the attack. Apart from that the small malicious activity can be conducted through legitimate requests, obfuscating any suspected withdrawal from the network. Therefore, the system is limited to the corporate network, and may only infer anomalous behavior based on the attack's partial traffic. Nevertheless, this is the main module and determines the source-end defense's success. The system in fullness must be aware of complexity of source-end detection. Since the results may have several levels of confidence, it is important that the response module be selective in that it must constrain outgoing traffic to the victim and preferentially treat legitimate traffic by sending it promptly while maintaining the provision of a first-class service. Thus, the source-end defense system should not only successfully detect and restrain many attacks, but also ensure the cross-cutting objectives of any defense system described in Section 2.6.4.

Finally, there is a dissent about the deployment of a source-end defense system. Corporations that deploy a source-end defense do not experience direct benefits since the system has an altruistic character in that it aims to prevent the occurrence of attacks originated anywhere from the organization's network, Internet misuse, and the attack's inherent consequences on all points of defense. Putting all the advantages in the other plate of the scale, the greater weight(i.e., main incentive) and motivation of this dissertation is to prevent the association and mainly the participation in DDoS attacks.

## 2.8 Generic Modules of a DDoS Source-End Defense System

Given the above discussion, with high ease of detection the reaction options are very limited, as in the victim side detection, where a simple load balancer health check is sufficient for detection. However, it is probable that there is a downtime in the services that the corporation exposes, resulting from the impossibility of applying more complex measures on the ongoing attack. On the other hand, with a greater complexity of detection, there is a whole set of options, with plenty of resources, to apply on the attacks in progress, easily preventing their arrival at the destination, including all the advantages in the detailed section above. On top of this, intending to make the most of all the advantages, it will be addressed the three modules that should be present on DDoS source-end defense system: monitoring, detecting and responding. The first module is important insofar as continuous network monitoring allows collecting information about the network that can be used as a target but also to initiate the attack through its nodes. In addition, constant traffic analysis enables the system to identify the services being used on the network, and map a usage profile that can converge to multiple levels(e.g., at user level, vlan, organization floor, ...). In this way it is possible to identify unauthorized services within the network. In order to optimize the monitoring process often instead of just one control point exiting the network, there is a segmentation within the network, distributing the processing capacity through the various strategically defined points. The main purpose of the detection module is to gather and analyze information from various

points within the network to identify possible security breaches, intrusion attempts, abuse of services or misuse, network failures, etc. The response module consists of procedures that are active to react according to previously detected failures and incidents, assuming the role of defense system spokesperson generating system responses that can range from just displaying an alert to messages to network administrator.

Network monitoring is a key point of every company, not only because it allows to manage an environment and to be aware of what is happening in the network, but also by the fact that, when acquiring data, it provides a work baseline for subsequent defense mechanisms. Mainly due to the constant growth of network data volume that travels, either within the enterprise and outside its borders, it is unpractical to a network sensor gather all the data. This data concept is wide, being the scope of the information defined by the sensor domain, presented bellow:

- **Network**

  Network sensors collects information about network traffic. Examples of these sensors include Netflow, most IDSs, raw data collected by *tcpdump*.
- **Host**

  Resides on the host and can monitor not only its activity but also the one on others(e.g., logins, logouts, file accesses, etc). Contrarily to network sensors, provide information on the low-level operation of a host, but is not able to gather information about the services that are running on a specific host. Host-based sensors are also limited to known devices, it is not possible to monitor unauthorized hosts. However on network domain, such communications are easily monitored.
- **Service**

  Service sensors are responsible for monitoring a particular service process, such as HTTP or SMTP server logs. There must be a previous study on the services running on a host to fully monitors all of them, maintaining track of all interactions, legitimates and malicious with such services.

There is always irrelevant information that can be avoided and, in order to obtain fully meaningful information, filters must be applied in the network layer, completely discarding the layers above, since trying to store all data referring to the header and payload of each packet is far too resource-intensive. Typically, in corporate network environments, the filter is applied on layer 4 of the OSI model, allowing to separate between the two transport protocols most used: TCP and UDP. In addition, another very recurring protocol in an enterprise domain is ICMP, which is often also taken into account when filtering information. Evidently, without depreciating the others, the representativeness of these protocols allows a reliable analysis based on extracted information such as:

- Packet's size;
- Source and destination IP addresses;
- For TCP or UDP traffic, the source and destination ports;
- Only for ICMP traffic, identify block connection attempts based on the content of Destination Unreachable;
- Only for TCP the sequence flags number.

#### 2.8.0.1 Approaches to monitoring a network

According to [120], there are two leading techniques to acquire data from a network environment: *active* and *passive* monitoring. The first mode, also called synthetic monitoring, is characterized by its real-time nature, and the injection of packets in the network that act as a probe to evaluate the network's performance, the available services as well as the network paths(e.g, packet delay, packet loss, routes, etc.). In contrast to the real-time, *passive* monitoring analyses the existing traffic over a arbitrary period of time and reports the results. Both modes have their strengths and weaknesses. Since the operation mode of synthetic monitoring is based on the introduction of a probe into the network, it introduces more traffic and therefore increases the load on the network hardware. On the other hand, allows collecting small amounts of data in real-time, measure traffic both inside and outside the network environment. *Passive* monitoring by not introducing any artificial traffic into the network, relies on historical and reports data based on a long period of time, providing a more holistic view of the network's health. It is also far less resource-intensive than active monitoring, albeit does not allow for measurements outside the corporate network, is limited to the device to which it is connected [121]. Another difference between the techniques is its scope: while synthetic monitoring manages to assure and test quality of service by generating network traffic, emulating various scenarios, passive monitoring due to its observational nature, and the fact that it uses large amounts of data allows predicting about bandwidth abusers using a standard inferred from based on historical data.

#### 2.8.0.2 Active Monitoring

According to [122], the most commonly used active monitoring techniques and tools to perform network measurements are:

- **Simple Network Management Protocol (SNMP):** is a popular protocol to active monitor network status. It follows an agent-manager architecture, with bidirectional communication between the *Network Management Systems (NMSs)* and *Agents*. An *agent* is a *managed device* which runs a software responsible to translates local management information data into a readable form for the NMSs(e.g., event and error information or performance information such as CPU usage). In its turn, these execute applications that monitor and control the managed devices( such as routers, switches, IP video cameras, printers, ...) that represent all the network's nodes from where the administrator or other software intends to obtain specific variables.
- **ICMP**: This protocol allows to reliably control and diagnose network elements, determining its availability and reachability besides estimating the delay and loss of packets.
- **Log data collection**: is the process of making sense out of the records generated by servers or devices in real-time, allowing to analyze its usage and respective performance, or even, with gathered information, simulate offline the user's activity in order to improve network's behavior.

#### 2.8.0.3 Passive Monitoring

Alternatively, passive monitoring does not inject traffic into the network. It operates quietly by collecting all the traffic that goes through a probe strategically placed in the network's topology. These probe solutions can be hardware-based, typically designated as Terminal Access Points (TAPs) or software-based.

**Hardware-based solutions**

A network TAP is a hardware component that connects directly into the cabling infrastructure in order to split or copy traffic for analysis and monitoring purposes. Due to its simplicity and passive-nature, as illustrated in Figure 2.12, is widely adopted on a variety of applications. For example, many organizations opt to tap all critical links for monitoring, security, or analytic use besides to provide easy access during troubleshooting or security breaches.

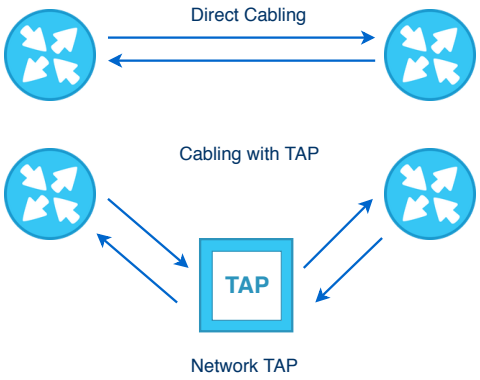



**Figure 2.12:** Direct cabling vs TAP operation mode.

There are two primary types of these solutions: passive and active TAPs. The main differences are related to the lack of a proper power supply, the signal retransmission and the device's interaction with the other network's nodes. Passive TAPs require no power source, and assume a completely discreet behavior on the network. These are neutral solutions which simply copy and retransmit the signal from one flow to a single switch port, using a built-in splitter mechanism. On the other hand, active TAPs interact directly with the connected components, by regenerating the signal which intercepts, for both destination retransmission and monitoring ports, avoiding split ratios present on passive taps [123]. However it becomes a point of failure due to possible power outages leaving it totally inoperative, creating a disruption in the network connectivity.

Given the absence of complications and the reliability that devices produce, TAPs are listed in most budgets related to the projection of a network architecture. Its usage constitutes the first step towards not only to any visibility solution and security analysis, but also to modern network-based IDSs [124].

Similar to the behavior of passive network TAPs, there is another solution that also mirrors traffic, called Switch Port Analyzer (SPAN). Widely known as *port mirroring* is a function of a switch or router that copies traffic from incoming or outgoing ports and forwards the

duplicated traffic to specific port - the mirror port. Comparatively to the previous solutions, it fails to aggregate traffic, since the sum of multi-port bandwidth exceeds that of the mirror port, contributing to the loss of packets and deteriorating the network element's performance.

**Software-based solutions**

Software solutions to monitor a network are even simpler, cheaper and easier to setup than the hardware ones. Usually, Operating Systems (OSs) come with CLI[36] tools, such as *tcpdump* which enables to capture, filter, display and save data about packets going in and out of an wired or wireless interface. Such simplicity enabled to developed arrays of monitoring software solutions on top of this packet analyzer. One acclaimed free and open-source example is the Wireshark, which allows analyzing network packets with aid of a Graphical User Interface (GUI) and a set integrated options(e.g., sorting and filtering), that allows network administrators to become aware of what is happening on the network at a microscopic level.

## 2.9 Machine Learning (ML) in Detection Methods

Every DDoS Defense system has built-in a network anomaly detection module which are, typically, based on a set of techniques that driven by the new attack vectors that have emerged in the recent past, have been incorporating novel machine learning approaches to the traditional measures. *Statistical* modeling is among the earliest methods used at level of Anomaly-based Intrusion Detection Systems (ABIDSs). Generally, a model that uses this approach begins by fitting a model for normal traffic so that it can then apply an inference test in order to confirm if a new instance belongs to the previously calculated model. All instances that do not conform to the learned model, based on applied statistical properties (e.g., mean and variance), are classified as anomalies. In addition to this approach, intrusion detection systems also rely on *knowledge-based* procedures. These are differentiated by the a priori necessary knowledge about attack signatures or other metrics, which are typically representations and general patterns of attack vectors formalized to identify subsequent occurrences. In this way, events or actions in the network are evaluated according to pre-defined rules or attack patterns to validate their legitimacy.

However, due to the paradigm shift, the detection of anomalies is aided by the machine learning in that it provides a broader view of network state, built through models that infer information from the network, such as normal and abnormal behavior along with other patterns that can not be perceived by the human eye. Although it is a concept currently in vogue due to the various areas where it provides help (i.e., image and speech recognition, fraud detection, health diagnosis), it is mistakenly to think that it is new, when in fact it is a concept already a few years old, and has been present, but with less regularity. For example, around the 1990s, was developed an Optical Character Recognitions (OCRs) system based on machine learning, which achieved a recognition ratio of close to 100% [125]. It is evident that today it is much more proliferated and the tendency is to continue, both in lateral growth

[36]Stands for command-line interface

and in improving the quality of people's life at various levels, from improvements in the area of medicine with standardized predictive models that can help cardiologists with specific guidelines for a particular patient [126], to improvements in traffic control in smart cities, optimizing their flow through data collected by IoT sensors and predictive models [127], [128]. Finally, what does ML means? According to Arthur Samuel, 1959, Ml is the "field of study that gives computers the ability to learn without being explicitly programmed" [129]. More elaborately, Tom Mitchell in 1997, defined as follows:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [130]

However, it is important to define the separation limit between ML and Artifficial Intelligence (AI) that are often confused being the same. AI is the broader concept that allows associating intelligence with machines due to the tasks they perform. ML in turn, is a subset of AI and its pillars are based on the idea of marrying algorithms and statistics and learning from the data in order to reach that intelligence [131], [132].

### 2.9.1 Traditional vs Machine Learning Approach

One of the most common ML usage examples are spam filters that aim to learn to flag spam given a set of spam emails (e.g., marked as spam by users) and examples of regular emails[37]. Using a traditional approach, statistical or knowledge-based, the detection module of the spam filter must follow the next steps:

1. Study the problem and recognize the most common spam patterns, such as the recurrence of words like "free", "amazing", "for you", etc.
2. Develop a detection algorithm containing rules for each of the studied patterns, which should flag emails as spam if verified
3. Test extensively, and add rules by repeating steps 1 and 2.

Figure 2.13 represents the conventional procedure to develop algorithms:

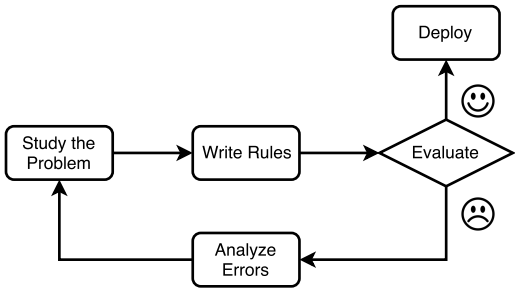



**Figure 2.13:** Traditional approach [129]

It is easy to conclude that the completeness of the system has flaws as well as the ability to scale and the flexibility to adapt to the new changes. In order to bypass this rules, spammers can easily replace common keywords, by unusual ones or simple acronyms of web culture, such

[37] Also refereed as nonspam or "ham"

as abbreviate "for you" to "4you" or even "4u". Each replication influences the spam patterns, which in turn need to be updated in order to keep up with the techniques's sophistication, resulting in an unending cycle and an increase in complexity proportional to the number of rules, reaching a point of maintenance impossibility. In a paradox depicted in the Figure 2.14, spam filters using a ML approach are more flexible, allowing for constant adaptation(i.e., progressive learning), recognizing new segments of sentences and words that represent new spam patterns, maintaining accuracy by flagging mails as spam or nonspam. In addition the complexity of maintaining the system is reduced, just as the general while the ratio of correctly[38] increases.

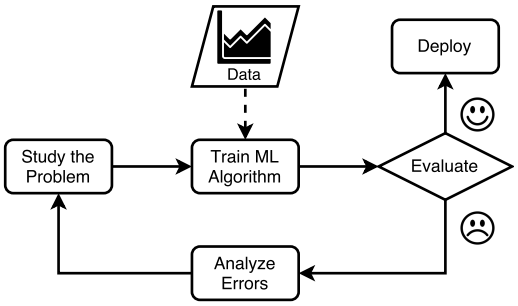



**Figure 2.14:** Machine Learning approach [129]

In these resolution, the system is able to learn from data examples called *training set*, so that it can later test with the next emails, each being a *training instance*. Another additional benefit is that the learning process from the training set and subsequent test with training instances can be automated, as illustrated in Figure 2.15, designing a quasi-automated system, requiring only monitoring.

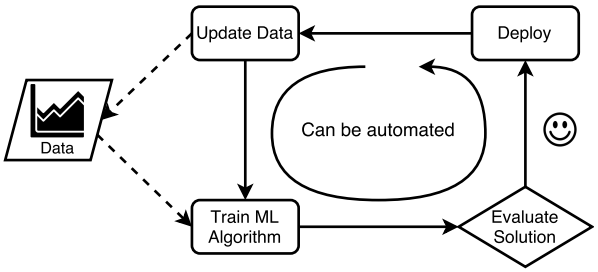



**Figure 2.15:** Autonomous Machine Learning approach [129]

Often, in complex problems requiring long lists of rules, handling a large volume of data, a continuous hand-tuning due to changes of general impetus or due to fluctuations of the environment, approaches based ML outperform traditional solutions such as Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) [133], [134].

[38] Also known as accuracy

### 2.9.2 Types of Machine Learning Systems

Since the learning domain in general is broad, there are several subfields from branches of the machine learning field that relate to specific characteristics of learning type. According to [129], it is possible to specify a taxonomy of learning paradigms:

- **Supervised/Unsupervised Learning**

  ML systems can be cataloged into four major categories according to the amount and type of human supervision required during the training phase: *Supervised learning*, *Unsupervised learning*, *Semi-supervised learning*, and *Reinforcement learning*.

  An exemplary task of *supervised learning* is *classification*. In these applications, in the training phase, each sample present in the *training set* is attached with a *label*, indicating the class belonging. Regarding network anomalies, the problem can be reduced to a classification issue, in order to distinguish legitimate from malicious traffic. Another common example includes the prediction of numerical values, such as the prediction of the value of a car given an associated *feature* set (mileage, make, model, engine size, interior style), also called *predictors*. In this context, the training phase of the *supervised learning* system includes data with *predictors* and respective *labels*, allowing to learn from the past, through a *regression*, to predict new values for new instances(i.e., new cars) [135]. Another important application of some regression algorithms concerns their use to classify, since their output indicates the probability of belonging to a class(e.g., 20% chance of a received packet to belong to the class of anomalies). The list of most important supervised learning algorithms includes [129], [136]:

  - K-Nearest Neighbors
  - Linear Regression
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Decision Trees and Random Forests
  - Neural Networks (NN)[39]
  - Gradient Boosting algorithms

  Antagonistically, in systems with *unsupervised learning* the training phase is equal to the nuance that each *training instance* is *unlabeled*, and the system attempts to learn in an autodidact manner, performing data *clustering*.

  Having as a frame of reference the banking institutions, through *clustering* it is possible to detect groups of similar transactions and conjecture about patterns. The result of this analysis may, for example, reveal fraudulent transactions, or in early stages may be able to remove these *outliers* preventing the occurrence of these anomalous incidents. Applications where data visualization is central also benefit from unsupervised learning. When the algorithms of these applications are faced with large volumes of data, complex and *unlabeled*, their representation in 2D or 3D and respective structuring and definition of limits can be aided by algorithms of *unsupervised learning* [129], [137].

[39] Some nn architectures can be unsupervised(e.g., restricted Boltzmann machines and autoencoders)

*Semi-supervised learning* occupies the middle ground, between the two extremes and represents the algorithms that deal with partially labeled data. Image recognition functions present in photo-sharing and hosting services like Google Photos are one of the use cases of algorithms of this nature. First, the unsupervised part is performed when grouping data possibly belonging to the same class(i.e., group the photos where a specific person shows). Then, a human labeling is required to identify each person and confirm the clustering process in order to determine everyone in subsequent photos [129]. *Reinforcement Learning*, on the other hand, is based on an entirely different paradigm. This approach is characterized by the presence of an *agent* that from observation of the environment and *policies* to perform actions will build and progressively improving their capabilities according to the reward or penalty of each action. It operates according to a paradigm of trial and error where the *agent* initially takes actions explicitly through *policies* to build a grounded base of actions with the respective rewards and, after executing all of them in an environment, begins to make decisions according to the most rewarding behaviors. Recently, a Japanese company *Fanuc* made an unusually clever industrial robot which just has to pick widgets out of one box and put them into another container. It is able to perform this task with any type of object, requiring only 8 hours[40], typically nocturnal, to reach an accuracy of 90%. It tries to pick objects while capturing images from the process, and each time it succeeds it remembers how the object looked, building its knowledge from these frames, to apply in the next morning [138].

- **Batch and Online Learning**

  Another principle that discerns ML systems is the ability to learn incrementally from new incoming data streams. Using a *batch learning* approach, the system is unable to learn incrementally and is therefore timely. Firstly, the system is subjected to a training phase in which it uses the whole *training set*, typically executed offline due to the need of computational resources, and after completion it transits to the production phase, where it applies the knowledge resulting from the previous phase. The limitation for the system to adapt to the new training sets implies that the system is replicated by another system that once learned from the new data, which is often unpractical and impossible to maintain the result of the slow learning phase and the need for resources for processing and storage.

  In contrast, *Online Learning* systems have the capacity to train incrementally, from the provision of new data present in the *training set* to the respective algorithms, in single instances or in small partitions called *mini-batches*. In relation to the batch approach it immediately provides advantages in the ease of training because it allows systems that do not fit into huge data sets to learn at their own pace by segmenting the *training set* into smaller sets that feed the algorithms. Another important advantage is related to time learning, since instead of being fully replicated the system has the ability to process the new *training instances* added to the *training set*, adapting quickly

[40] Eight robots working together for one hour can perform the same learning as one machine going for eight hours

to the fluctuations of the environment. However, one must pay attention to the *learning rate* parameter present approach, which as the name refers, indicates how quickly the system accommodates the changes. When this parameter assumes high values, the system adapts instantly to the changes, quickly forgetting the first data learned. This immediately raises a problem of gradual decline of accuracy, because in cases where the data that feed the algorithms are incorrect, the ML system will tend to lower its performance. In contrast, systems with a low *learning rate* will be more tolerant to faulty data yet will be less adaptable to environmental transformations at the same time.

Thusly, online learning draws attention to keep the system's performance from time to time, mainly monitoring the changes in the *training set*, since when improperly altered it can indicate an incorrect rhumb in the system, being necessary the revertion for the state previous [129].

- **Instance-Based Versus Model-Based Learning**

  As the main objective of many ML systems is the ability to predict, they are divided by how the prediction is grounded since it directly influences the prediction's accuracy and how the system can generalize to new data. Conforming to [129], there are two main approaches to generalization: *instance-based learning*[41] and *model-based learning*. The first technique follows the form of learning more trivial: learn by heart. The system is able to predict by comparing new instances with those memorized during training. Although the complexity of classifying a single instance is directly related to the size of the training data [139], the system has the ability to generalize its model to previously unseen data through *measures of similarity*[42] between instances. On the other hand, the *model-based* system based on the training set creates a model which is used later in order to make predictions through a regression. On top of this, the system provides performance measures, such as fitness and cost functions [129], which define how well and poorly the new instances fit the model, allowing a quantitative evaluation between the prediction and the current result.

### 2.9.3 Main Challenges of Machine Learning

From a high perspective, a typical machine learning problem includes the agreement of an algorithm and its training with a set of data. Thus, only two aspects can become objections to the problem: the algorithm or the data.

#### 2.9.3.1 Insufficient Quantity of Training Data

A famous study by two Microsoft researchers, Michele Banko and Eric Brill in 2001 revealed a direct proportionality between the volume of data and the efficacy of the natural language

[41] Also known as memory-based learning
[42] Typically context-related statistical metrics

disambiguation problem[43]. The author emphasizes this dependence suggesting that the trade-off between spending time and money on algorithm development or spending it on corpus development should be rethought [140].

So, given a specific context and an algorithm how many samples are needed? The answer is: it depends. There is no strict rule, it depends on the problem's complexity and the learning algorithm's complexity. In order to evaluate the algorithm's performance as the number of *training instances* increases, there is a heuristic called *learning curve* that refers to the relation *prediction accuracy/error* versus *training set* size [141].

#### 2.9.3.2 Non-representative Training Data

Generalization is a key part of machine learning algorithms since it directly influences performance. Thus, it is crucial that the *training set* samples represent the cases to be generalized. This requirement applies to both *instance-based* and *model-based* systems. A famous example of sampling bias occurred during preparation for the 1936 presidential election, which placed Franklin Delano Roosevelt against Alfred Landon. As always, a number of polls were conducted, including a very peculiar one belonging to Literary Digest magazine, which stated that more than 2.4 million respondents had intended to vote for Alfred Landon. However Roosevelt won 46 of 48 states, which completely disproved the poll that Landon would win by a landslide. Subsequently it was discovered that the magazine probed its readership which were skewed in a subgroup that supported Landon. Thus, despite the large sample size, the data were not representative at all, preventing subsequent generalization from being correct [142].

When it comes to training it is important to understand that large sample volumes do not mean that *training instances* can not be biased. Particularly now with the trends of "Big Data" and Data Mining (DM), the *training instances* must be filtered so that they are representative for the data that is intended to generalize.

#### 2.9.3.3 Poor-Quality Data

Similarly to non-representative data, it is also possible to infer a relationship between the number of errors, outliers and noise, and the system's probability being unsuccessful along with the subsequent difficulty of detecting reliable patterns. In order to reverse the relationship, the training set should be cleaned, discarding all samples with clear outliers or noise. The result will be a smaller *training set*, but more valuable at the expense of noise. Another important aspect is the samples's completeness in the *training set*. When some features are faulty, it is necessary to choose wisely if it is possible to discard the instance or if the missing features are to be filled, being aware of the influence on the model. Thus, it is preferable to spend time once on the pre-training algorithms by cleaning the input data instead of continuously expending in the post-training trying to calibrate and adjust the algorithm behavior [129].

[43]For instance, confusion sets include: ("principle, principal"), ("to,two,too"), ("weather,whether"), depending on the context

#### 2.9.3.4 Irrelevant Features

*Feature engineering* is a key process in ML algorithms, since it is the algorithm book, and they can only learn if the book is appropriate. If the *features* present in the *training set* can not model the problem(i.e., lack of relevant features), it is clear that the result will not be as expected, it is impossible to learn mathematics by reading geography books. *Feature engineering* is a process which encapsulates another three [129]:

- *Feature Selection*: Select the most useful and significant *features*
- *Feature Extraction*: Process that aims to produce more beneficial and relevant *features* from those previously selected
- Gather new data progressively so as to continuously produce new *features* capable of generalizing the problem.

#### 2.9.3.5 Over-fitting and Under-fitting the Training Data

Regarding the choice of algorithm, it is vital to understand how the algorithm learns by analyzing the *fitness function*, predicting that the system falls into the overgeneralize trap. This happens when the system fits perfectly in the training data, but then to fail roundly for new predictions, falling into *over-fitting* [143], [144]. Paradoxically there is the concept of *under-fitting*: it occurs when the ML algorithm cannot cannot capture the underlying trend of the data(i.e., the model is too simple). Both *over-fitting* and *under-fitting* lead to poor predictions on new samples. In order to prevent the occurrence of these phenomena, the first step is to select a powerful model with more parameters. Subsequently, at the level of *features engineering*, measures should be adopted to improve features and reduce regularization [129].

#### 2.9.3.6 Testing and Validating

In order to evaluate how the model generalizes, it is necessary to submit it not only with *training set* samples but also new test instances. Typically, the *training set* is branched into 2 sets: *training set* and *test set*[44].

When it comes to testing the system with *test set* samples, the resulting evaluation of system performance that indicates how well the system responds to instances never seen before is mapped into the *generalization error*[45]. This also provides relevant information when comparing two models with a similar behavior, thus acting as a selection factor alongside that can also be used as a parameter calibrator [145]. For instance, when the *generalization error* assumes high values, it means that the model has fallen into *over-fitting*. To avoid this incident, it is common to use the cross-validation technique: the *training set* is partitioned into complementary subsets that combined differently to allow for generalization. Nonetheless, at each iteration, the remaining portion of data that is not used for training is used for validation(i.e., called *validation set*), providing an accurate *generalization error*. Afterwards, the final model is submitted to a final validation, facing a full set of new instances from

[44]Commonly 80% for training and hold out 20% for testing
[45]Also known as "out-of-sample error"

the *test set*, allowing an evaluation of the model's generalization [129]. Finally, given a set of data which model best fits? *No Free Lunch Theorem* [146] is often used as an example. Since a model is a simplified version of observations, without superfluous details, when no assumption is made of the data, there is no reason to prefer one model over any other. There is no guarantee *à priori* that a model works best for a given data set and therefore is chosen over another in future cases. The only way to know which one is best for a particular case is to evaluate them all[46].

## 2.10 State of the Art

Over the years, awareness has gone hand in hand with the development of new solutions to combat and mitigate DDoS attacks. Today, these are present on every security mechanisms implemented at corporate network level, being a major component when it comes to detect and react to egress DDoS attacks. The described solutions to detect abnormal behaviors are divided into non-commercial (i.e., implementations presented on conference papers and articles) and commercial systems, products sold to the business sector, abstracting implementation details for the general public.

### 2.10.1 Non-commercial solutions

Initially, the first resolutions to detect and prevent DDoS attacks at the source, appeared in publications of academic and research dissertations and articles. Chronologically, in 2001 Gil et al. [147] proposed a solution, designated as *MULTOPS*[47], that monitor traffic characteristics of network devices, such as routers, which maintain a data-structure containing packet rate statistics for subnet prefixes at different aggregation levels. In order to detect bandwidth attacks, the solution was based on the assumption, between two distinct networks, that the incoming packet rate is equal to the packet rate that exits during normal conditions. Thus, a network device using *MULTOPS* it would detect bandwidth attacks when there is a significant unbalanced in the packet rate to and from a target. However, the detection system has some limitations mentioned by the authors. In order to bypass the detection system, the attack vector just needs to randomized the IP source of malicious traffic. Besides, the large number of agents results in small traffic rate for each one, which make the flow proportional affecting the system's ability to detect low-rate attack traffic. Furthermore, *MULTOPS* suffer from a high positive rate when faces streaming services once its flows are disproportional. Two years later, Mirkovic presented his PhD dissertation [10] and lately a white paper [148], where he developed a system comparing inbound and outbound traffic on the source side to detect DDoS attacks. The system, called*D-WARD*, followed a statistical approach, which based on statistics extracted from bidirectional traffic in periodic deviation analyzes with normal flow patterns allowed the detection and significant reduction of attack traffic. In order to bypass this detection method, attackers just have to adjust their traffic dynamics according to the

[46] It is possible to make some assumptions to restrict the set of algorithms to be tested
[47] Stands for MUlti-Level Tree for Online Packet Statistics

normal profile. In 2006, Sekar et al. [149] proposed a triggered, multistage approach designated as *LADS*(Large-scale Automated DDoS detection System), which were mainly characterized by its scalability and accuracy. The system was modulated to be deployed at the ISP level, acquiring data through protocols such as SNMP and Netflow[48]. In a first phase the SNMP data is analyzed and if metrics of interest are found, such as bytes or packets per second, is triggered an alarm that collects data through Netflow in the router and specific interface for a more granulated analysis. On this phase, is applied a unidirectional clustering based on a set of thresholds for each metric, as described in. Six years later, another solution, deployed at ISP level, was proposed by François [149]. Known as *FireCol*, relies on a distributed architecture composed of multiple ISPs forming overlay networks of protection rings. This barrier is created around the host/network that needs protection, when each surrounding ISP installs the IPS, which is provided by *FireCol*, allowing the system to communicate horizontally and pass an alarm or some statistics related to packet rate and overall bandwidth. Four years ago, an interesting approach was proposed by Ivo et al. [150]. Although the detection model depends on the periodicity inherent in the attack traffic, it would detect and quantify DDoS attacks at the source, even when performed using encrypted channels or obfuscated in legit traffic. The detection method uses multiscaling traffic analysis based on wavelet decomposition through the Continuous Wavelet Transform (CWT).

Although these methods are particularly designed to be implemented at the source, there are others implementations with interesting features that can be moved away from the target. Due to the growth not only of the cloud environment but also of the areas related to machine learning, quite a few solutions have been emerging recently based on automatic learning for these scenarios. In the last year, a DDoS detection system [151] was designed for these scenarios, aiding both signature detection techniques(i.e. *Snort*) and ML algorithms, including decision trees generated form the *C4.5* algorithm [152], K-means and Naive Bayesian. The dataset was acquired from a implemented setup, where the normal traffic was generated through simple scripts while attack one was obtained from *hping3*, a command-line oriented TCP/IP packet generator mainly present on Kali Linux [153]. The focus was on flooding-based attack targeting layer 3 and 4 in the OSI model. Despite the good results and the possible transaction from a cloud to a corporate environment to detect attacks coming from the internal network, the solution sins in the number of attacks that it is prepared to detect, being in an unfavorable position to detect attacks of the other two categories or even zero-day attacks.

### 2.10.2 Commercial solutions

Although none of the previous solutions have been commercialized, it has helped to conserve the constant threat of cyber crime and the development of new solutions. Now, with constant and powerful threats comes respect and with it prevention. A major difference from the non-commercial solutions are the implementations details. So the only way to gain a comprehensive view of each solution is based on reviews from other companies or through classifications

[48]Is a protocol developed by Cisco to collect and monitor network traffic

based on use cases.

Actually, each Internet's giant provides solutions for corporate environments, ensuring its security, whether for internal attacks or for external attacks, providing reliable and resilient mechanisms to keep all the core services up, as long as possible. Although it is not a limited solution to detect DoS attacks coming from the internal network, it contains several features that allow companies to obtain direct benefits of their use, mainly guaranteeing protection against external attacks or data exfiltration. Of note the top five, according to [154], which is composed of:

- Imperva Incapsula
- Cloudflare
- Arbor DDoS
- Sucuri
- Radware DefensePro

Cisco also provides a similar system known as Cisco *Stealthwatch*, which comes in $17^{\text{th}}$ on the ranking, and like the previous commercial solutions include features scalable visibility, security analytics, advanced threat detection, accelerated threat response and simplified network segmentation [155]. The system is able to trigger an alarm based on suspicious events, such as:

- Source or target of malicious behavior : scanning, abusive network activity such as file copying or transfer, policy violation, etc.
- Reconnaissance : port scanning for vulnerabilities or abnormal running services
- C&C : Communication back to a master server through malware
- DDoS activity : send or receive data floods
- Insider threats : data exfiltration

This system uses cognitive analysis, a cloud-based threat detection and analytics capability, where a baseline profile is assembled, from Netflow, IPFIX and other types of network telemetry, allowing to apply context-aware analysis to automatically detect anomalous behaviors. In addition, switching core processing to cloud computing is a huge advantage, allowing to achieve a holistic view of network traffic and general packet analysis(i.e., local and global). Thus, ML algorithms are applied over data collected, including encrypted traffic, enabling a continuous monitoring, analysis, and responses in time.

In conclusion, DDoS attacks have attracted a lot of attention in research and commercial communities, and today's corporations look to the these as a major threat. History proves the growth of attacks, not only in scale but also in frequency. It is easy to conclude that even the modern corporate networks can be easily a target, and it is extremely difficult, on the victim side, to react quickly and effectively, ensuring that there is no downtime for the services that the company exposes. Although this is the typical scenario, it is possible to be changed in the future by adopting source-based detection solutions. Despite the greater difficulty of detection, it is possible to avoid this scenario being repeated, by adopting the previously detailed requirements, and to include the use of machine learning, making use of the increasing amount of traffic currently circulating in companies, to achieve reliable results and times decisions.

CHAPTER 3

# Methodology for outbound anomalies detection

*In this chapter is discussed the methodologies to infer network attack patterns. The whole process is iterative, resembling an assembly line, where the raw data from the network is the input, whereas the output contains the decision about the normality of a network observation that reflects a behavior. The chapter is organized sequentially according to the assembly line, where the first section details the chain of how data must be acquired, processed and stored according to a data structure that aims at easy manipulation and, ultimately, the generation of observations that map a set of actions. It also included a generic approach to the overview and analysis of the various network observations, and finally the classification and evaluation process. Overall, it describes the process from the data preparation to the use of ML to detect patterns.*

## 3.1 Network Data Collection

As discussed in Chapter 2 the prime concept is to infer attack patterns supported by a set of observations that generalize network activity. In the absence of monitoring protocols, such as SNMP or Netflow, and independently of the network environment, its control is solely based on raw traffic analysis. However, the network environment concept is wide besides the diversity of entities that are present, specially at organization domain. As explained in Section 2.8 this dissertation is confined to network domain, however when it comes to collect data at this level, there must be a balancing between completeness and redundancy. For example, monitoring communication between two hosts belonging to different VLANs[1], should be accomplished by inserting a network TAP between the access and distribution layer, among switch and router respectively. By moving network TAP to lower level, at a switch interface, besides the inter-communications also internal one between same VLANs,

[1]Stands for Virtual Local Area Network

would be collected, resulting much useless data. The implementation of both approaches result in redundant data, since same traffic would be gathered in different points replicating information about an event. Furthermore, increased cost and maintenance should also be addressed when placing the sensor in higher layers. There is a greater complexity to monitor the greater the distance to the access layer, where normally all users are present. Shifting the purpose to a malware propagation context, adopting a SPAN solution at access layer level would be the best approach, since such malwares tend to spread horizontally by compromising nearby devices, typically on same VLAN or subnet. Therefore, place the sensor at higher layers on the network's topology, for example at core-level, result in a higher host's abstraction due to the distance and impossibility to closely analyses their actions, since many of these does not reach higher layers.

Concerning the requirements, when the problem's scope focuses on anomaly detection generated from inside to outside, the ideal sensor location is in the edge, at core layer separating the corporate network and Internet. Every connection to outside the company's network are necessarily intersected by this layer. Since intra-communications does not represent useful information, only a single hardware or software-based approach is required to gather all gross outbound traffic.

Due to packet's stack nature, its examination yields more detailed information the larger the boundary layer. However, this should not be arbitrary and in order to prevent legal consequences from possible privacy invasion claims, the analysis must be limited by the OSI's model fourth layer. Storing only four layers per packet represents a significantly optimization, nevertheless, it is necessary to disseminate those. There are a lot of control fields which does not represent any meaningful information and should be discarded, leading to quasi optimal approach in which only the data obtained from a chosen model are saved. Generally, up to the fourth layer, the relevant information is portrayed in Figure 3.1.

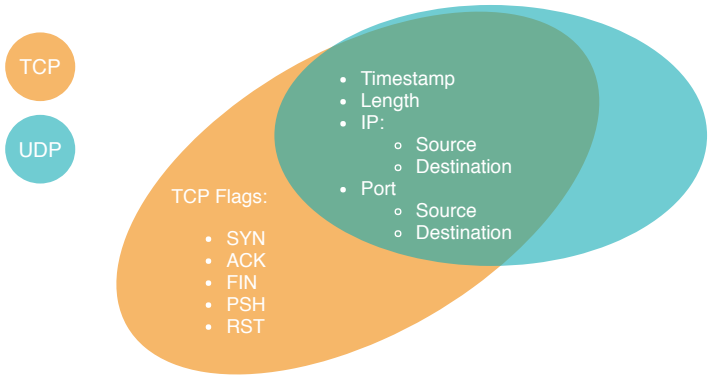



**Figure 3.1:** Metadata collected from the network traffic and how packets fields from different protocols are stacked.

Each packet field contains the following information:

- Present in TCP, UDP:
  - Timestamp: packet's UNIX epoch time,
  - IPSs: source and destination IP addresses,
  - Length: packet's size,
  - Ports: source and destination ports;
- Only present in TCP packets:
  - TCP Flags:
    - SYN: If synchronize flag is set,
    - ACK: If acknowledge flag is set,
    - FIN: If fin flag is set,
    - PSH: If push flag is set,
    - RST: If reset flag is set;

## 3.2 Features Engineering

### 3.2.1 Packet fields to features

The metadata acquired by the passive monitoring module per se does not represent any information that describes a pattern or behavior. However, this agglomeration of fields represent the necessary foundations to obtain features, some simpler and more direct than others, that map a set of actions on the network. From a high level, the variation of network activity is defined not only by the entities's behavior that compose it, but also by their number. These may assume divergent behaviors, which vary not only in frequency but also in scale.

In this way, the approach of mapping packet data over time instantly emerges. For instance, based on a day periodic scale, and considering the context of a banking agency, the volume of traffic between an ordinary employee who is only present during daytime is different from a compromised device running 24 hours a day, which attacks multiple targets. On the other hand, assuming a smaller scale, at the minute level and considering the use case of watching a Youtube video, its beginning is characterized by a series of bursts until all the video is loaded, and then a long silent period. At this level, recurring events produce periodic packet rates which allows to identify the underlying services. However, this is not always straightforward. In addition to the fluctuation along time, there is also no rigor as to the frequency and nature of the events, thus are defined as pseudo-periodical, but not vehemently. These events variations reflect, what is happening on the network, and once it is a low-level representation of it, there must be right balance between detect such events and tolerate these variations.

Thus, not only for temporal dependence, but also to facilitate pattern recognition, packet data must be indexed over time. Considering a simple capture that occurred during a certain period of time X, in a network interface, a possible approach to represent its events over the time is illustrated in Figure 3.2. In an anomaly detection context, instead of contemplating the capture as a whole, by analyzing it from the left side to the right, it is preferable to continuously

analyze several time-equally periods[2], which contain statistical descriptors obtained during that smaller period.

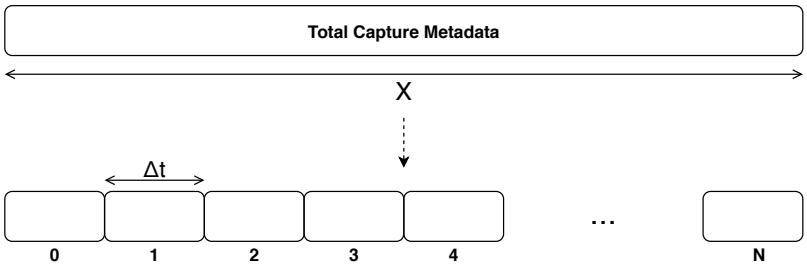



**Figure 3.2:** Partitioning in smaller sampling windows that contain statistical metrics obtained from the metadata during a $\Delta$t time frame. The total number of sampling windows is defined by N = X $\div$ $\Delta$t, X $\geq\Delta$t.

There is a direct relationship between the capture's length and the sampling windows number. For instance, considering a one-day capture, a single sampling window should be adopted when it is intended to detect and infer patterns that occur and alternate at a 24-hour scale(e.g., weekdays pattern, weekend, public holidays, holidays, etc.). In its turn, 24 sampling windows, resultant from a division by one-hour, would allow to deduce employees' work hours and respective breaks. These variations on the window sizes pose a major factor because allow the detection of patterns at various scales and scopes.

Each sampling window with $\Delta$t length, contains a set of metric statistics that are purely metadata sums over a time frame. This *statistical processing* operation is abstracted from the packets nature and represents the first step to modeling a pattern using several time-dependent attributes:

- Statistical packets counts

  Metrics based on packet counts and their attributes formalize the first step in defining the action pattern of each attack(e.g,. bytes sent and received).
- Network protocols usage

  Packet counts per protocol, and inherent percent utilization, represent a pattern allowing, for example, to infer its position in the network topology.
- Data streams aggregation

  Communications between entities is represented by data streams, which allow to detail its duration, edge entities, total bytes and so on. For instance, in a typical syn flood scenario, aggregating traffic by data streams allows to demonstrate a huge amount of syn flags in one stream over the others, as well as the ratios discrepancy between streams.

  Since a service is mainly characterized by an exposed port, it is also possible to analyze the number of sessions and the host that uses a particular service more often.

When the scope focus on modeling an attack, these are the direct attributes, however, depending on the context and inherent patterns that should be modeled, the list could be extended. At this point, one must understand the relation between the sampling window's

[2]Also known as sampling window

duration and the abstraction it represents through the various attributes that constitute it. Lower $\Delta$t means less data is aggregated and therefore high-frequency events are easily detected, such as host-to-host communications due to its occurrence on small time scales. On the other hand, larger time frames ($\Delta$t), allow to combine more metadata and thus detect patterns occurring at temporal scales similar to human actions.

This process constitutes the second node in the network observations production chain, being the first represented by data acquisition and packet's data filtering. Thus, statistical descriptors will be later used to produce features and these ultimately, will generate network observations. However, in the absence of subsequent nodes, it is possible to make decisions based on these metrics and thresholds obtained based on historic of such statistics. In systems that follow this approach, these decisions represent another constraint in choosing $\Delta$t value of each sampling window. The decision is taken after each $\Delta$t, however, lower time frames may not allow finding deviations once gathered metadata may not be representative although allow acting quickly. Contrarily, higher values of $\Delta$t allows to summarize metrics for a longer period of time, however the action is taken later, and in some scenarios it may be too late, as in data exfiltration contexts. Thus, it is necessary to find a balance between the response time and the representativeness according to the scope of the anomalies.

This prototype structure ensures not only the simplification and aggregation of data but also its safeguarding in a simple arrangement of storing and iterating over it. Nonetheless, the inherent *statistical processing* operation requires computational resources, and ideally, at production environments, should be performed right after the metadata filtering, storing only a simplistic structure, as illustrate Figure 3.3. On the other hand, at development environments, these two stages should be decoupled enabling data analyzing and manipulation on both levels.

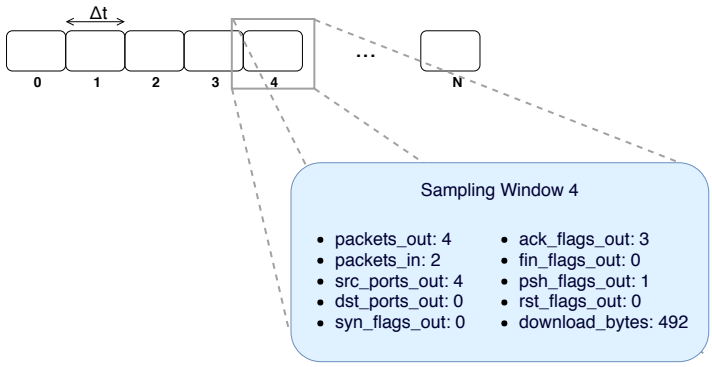



**Figure 3.3:** Low-level structure of a sampling window.

Another import concept is the structure's organization and its growth. Considering an initial sampling window with a duration of one day, its division in 1-minute windows(1440) in which each reserve 10 attributes, results in a bidimensional array of (1440, 10)[3]. Its replication over a whole week generates a tridimensional array of (7, 1440, 10) producing a total of 100800 values which represent the host's activity on the network for a week. Generically, when each

[3] **Also known as matrix**.

sampling window's length is lesser than one day, the tridimensional array is defined by the following tuple: (days, sampling windows number, total attributes per window).

### 3.2.2 Generating network observations

Although chain's second stage is fundamental, the result of *statistical processing* operation does not necessarily model entities activity at such level that could be inferred patterns. It is a low-level computation that simply aggregates and sums metadata, retrieved from the first stage, following a criterion defined by the detection's scope. Thus, these statistical metrics needs to be processed and transformed into something that actually model an attack behavior, allowing to perceive patterns changes. This represents the last stage of the chain whose objective is the production of network observations, serving as input to the data pipeline, which contains at the last stage ML algorithms responsible to detect patterns and produce answers.

The sequential sampling window paradigm can be transposed to a possible third stage. At this one, there is a sequential iterative process, where at each iteration results a new observation window with features computed from the various attribute sets. Therefore, each new observation window is a macro-view over the second stage resultant windows. As Figure 3.4 shows, this initial third stage approach is extended to previous described example's visualization.

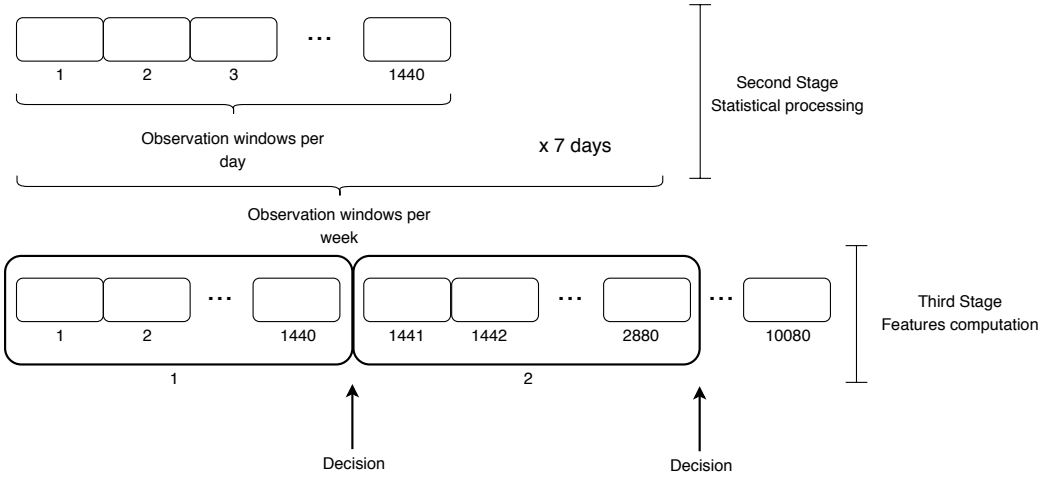



**Figure 3.4:** Second and a proposed third stage, aiming to produce network observations based on the host behavior over a week. The observation windows, at this third stage, have $\Delta$t value of 1 day.

Based on the example's context it is possible to describe three correlations due to the subsequent architecture, between the third stage and the data pipeline:

- Modeling process and ML algorithm's results

  The first correlation concerns the system's ability to produce better responses. Thus, improvements in the third stage, particularly in the ability to generate network observations that effectively model and have an explanatory character, are reflected in relevant and easy-to-obtain results.

- Observation window's duration and response time

  There is a high dependency degree between network observations time's duration and the response time, since defining one determines the other. According to the example, assuming a $\Delta$t value of 1 day, implies obtaining the representation of a whole day data, in order to confirm its normality or abnormality, and solely after day ends is possible to fetch a response. On the other hand, faster response times can be adopted, however, it must be kept in mind that low $\Delta$t values may mean non-representative data, since the patterns may be dissolved throughout the various observations. These contain very similar data that do not allow the perception of pattern changes.
- Observation window's length and network observations number

  The number of network's observations can be defined through a function that varies the output with respect to the duration of each window, as shown in the following Table 3.1:

**Table 3.1:** Relation between observation window's length and number of observations generated. It follows the previous example, where sampling windows assumed a $\Delta$t of one minute and the capture total length is one week indexed on 10080 sampling windows.

| **Observation's length** | **Windows aggregated** | **Observations number** |
|---|---|---|
| 5 minutes | 5 | 2016 |
| 10 minutes | 10 | 1008 |
| 30 minutes | 30 | 336 |
| 1 hour | 60 | 168 |
| 5 hours | 300 | 33 |
| 15 hours | 900 | 11 |
| 24 hours | 1440 | 7 |

The granularity of each observation window is defined by the system's objective. For example, analyses a whole day in hourly sequences may be right decision to infer consecutive low-level DDoS threats or large deviations in the average amount of bytes leaving the network, indicating the leakage of information.

Thus, the resulting number of network observations may be a constraint to the ML algorithms, since some need a minimum of observations to produce reliable results. Therefore, it is important to find a balance the system's response time and number of observations needed, defining this way the observation window's length.

With these premises in mind, a new approach emerges, shifting from a sequential and strict paradigm to a slider. As Figure 3.5 illustrates, an observation window is superimposed over the resulting second stage sampling windows, which slides to the end. In each iteration computes features based on the metrics present in the sampling windows it covers.

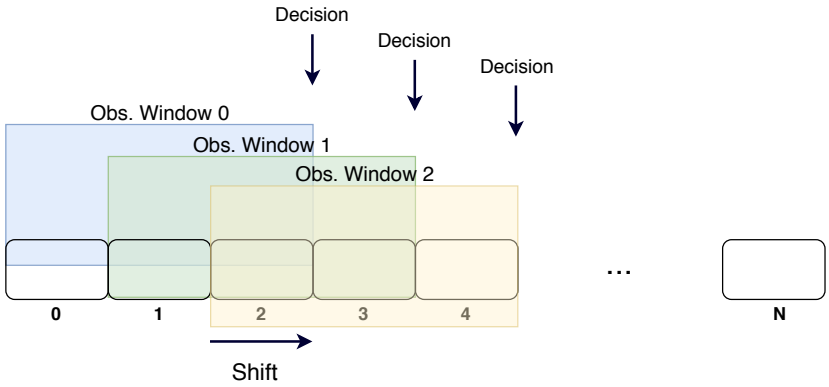



**Figure 3.5:** Observation window horizontal movement throughout the various sampling windows. The shift time is equal to sampling window's length.

The sliding window iteration strategy has various advantages. The resulting number of observations increases immediately due to the sliding character. On the previous approach, a set of aggregated windows were exclusive to a network observation, which is not verified on this one, since consecutive iterations coverage a portion of equal sampling windows. Thus, behaviors that slide through the day, such as compensating hours at an unusual time, are deliberate as normal, rather than the previous solution that due to the deep connection of the action with the day's period, would consider as abnormal. Another important advantage is the significant reduction in response time. This is defined by the window's step time, in detail: the response time in the first iteration assumes the observation window's duration, as in the previous solution, and in subsequent iterations it decreases to the shift time.

Thus, as shown in the Table 3.2, different window sizes generate different number of observations and therefore, as in the previous solution, the right balance between the various free parameters must be found, in order to detect events and patterns that occur in a long or short period of time.

**Table 3.2:** Number of observations produced from varying observation window's size and sliding window shift time. Values are also based on the previous example.

| Observation's duration | Shift time | Observations number |
|---|---|---|
| 5 minutes | 20 seconds | 30226 |
| 10 minutes | 40 seconds | 15106 |
| 30 minutes | 5 minutes | 2011 |
| 1 hour | 10 minutes | 1003 |
| 5 hours | 20 minutes | 490 |
| 15 hours | 1 hour | 154 |
| 24 hours | 6 hours | 25 |

Likewise, the sliding window's decision size is taken depending on the system's scope. For example, sliding windows smaller than a minute wouldn't be able to detect Youtube activity due to its pseudo-periodicity that occurs in the order of few minutes. At the other extreme, a hourly periodic event(e.g., such as backups in a production environment) would be detected

considering a sliding window's size of 24 hours.

### 3.2.3 Network modeling

After making the decision about the sliding window's size it is necessary to transform the various attributes, present in the aggregated sampling windows, in features and consequently, network observations. In each iteration, it is possible to construct an bidimensional array, represented in Figure 3.6, with the various values of each attribute present in the aggregated windows. Therefore, the number of columns is equal to the number of attributes and the lines number is defined by the sampling windows combined, which is always equal in all iterations.

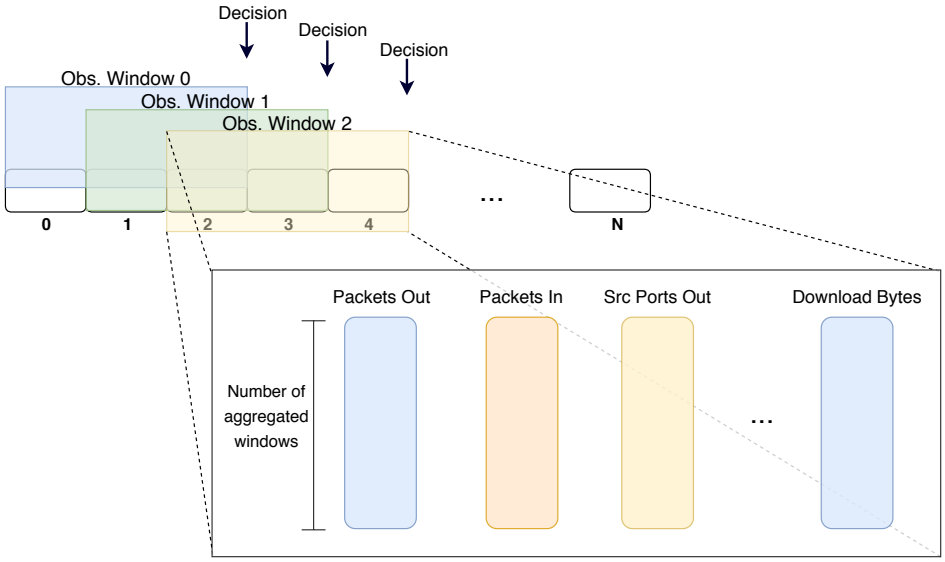



**Figure 3.6:** Representation of sliding window paradigm, where at each iteration shifts one minute(i.e., duration of sampling windows), overlapping two windows after the first iteration.

A possible approach to compute resources arises from the interpretation of the various matrix's values so that several time-independent descriptive statistics(e.g., such as mean, variance, median, quantiles, etc.) are produced from the same attribute. In addition to these features, it is possible to select the zeros of an attribute, deriving new features, being these ones time-dependent. For instance, considering the packets number, the zeros, along the column represent passivity in the network and from these it is possible to define the average time of a silence period. Usually, as in the purpose of the example, this method is applicable to attributes that are related with network's silences periods being also possible to extend the method's scope to a whole window, where are counted all the zeros from all columns, resulting a new features which gives an insight about the window's completeness.

Thus, the chain cycle ends resulting in a set of network observations with descriptive statistics that discern patterns, and since the algorithms are particularly optimal in this area, these observations represent the various data pipeline's input samples.

### 3.2.4 Features that describe attack behavior

In order to model a attack vector behavior it is necessary to understand how is performed along with the traffic that is generated. From these, and due attack vectors diversity, should be selected unique sets of features that characterize each vector, since it is impossible to address a single set that define all the patterns. For example, to model a TCP STOMP flood, would be useful to determine features such as number of TCP session, number of ack and push flags along with its ratio per TCP session. However, when performed in a file sharing context, those may be not sufficient to model it, since in both cases these features assumes similar values preventing patterns differentiation.

Another metric that potentially separates attack from human's traffic is its rate and periodicity. Attack vectors that assume human dynamics increases exponentially the difficulty to distinguish patterns, unless the traffic generated highlights completely from the normal one. For example, considering a network with several employees that just browse and listen Spotify, even if a smurf attack follow employee's dynamics, would be easily detected due to the abrupt changes, once at normal conditions there are no ICMP traveling on the network. Relatively to event's periodicity, at a small temporal scale, humans tend to have higher behavioral variations that are reflected in aperiodic actions. In its turn, actual attack's dynamics, independently of its rate, manifest a periodicity over the time, which contrast with human's activities at same scale.

Based on this assumptions, the feature selection to describe a behavior is an experimental and iterative process from where should result statistical values enabling to discriminate normal from abnormal activities. However, after its selection it may result poor-quality data that needs to be analyzed/filtered in order to obtain a completeness set which will produce better results.

### 3.2.5 Features completeness

Not always more features mean increase of performance, in fact, in most cases the result is not the conjectured, lowering the accuracy and slowing the training phase due to computational complexity inherent to excessive number of features. On the other hand there are no rules to define the ideal features number in a generic classification problem. It is only possible to address a dependency between the features number and the samples. The lower the features number less samples are required for the model to generalize correctly. On the other hand, the number of samples needed grows exponentially with increasing features. This problem is often referred to as the *curse of dimensionality*. Transposing to a simple example, it is easy to find a certain number in $10^1$ numbers. However if the search space's size is increased to $10^2$, the difficulty degree increases exponentially and so on. This exponential growth in data causes high sparsity, and since it is impossible to achieve an infinite samples number to combat the increase of feature dimensionality, it is important to understand that for N number of features should be such that *feature dimensionality* $\ll$ *training samples*.

Thus selecting the optimal network observations number for N dimensions is a major

concern on classification problems. In order to simplify this process there are two recurrent ways: *feature selection* algorithms are the first approach, which employ heuristics(e.g., greedy methods, best-first methods, etc.) to find the best combination and feature number. In its turn, *feature extraction* methods are the second and most notable approach to find the optimal a combination of original features, reducing the final problem's dimensionality due to the transformation from N original features to M, where $M \leq N$. A well known dimensionality reduction technique is Principal Component Analysis (PCA).

## 3.3 High Level Data Overview

### 3.3.1 Dataset analysis

So far the only view that can be drawn from the data is light because of the inherent low-level process. As previously described in Section 3.2, from the transformation of the raw data into sampling windows with metadata and from these to network observations that model the pattern of an activity, allowing to infer about its normality, the general comprehensiveness of the data type being treated is nil and, it is only possible to gain insights from the data by analyzing network observations.

A good starting point starts with the analysis of the dataset following an approach that must start from a higher level to the lowest one, where the dataset is initially interpreted as a whole, segregating later for each label or component. Initially, the samples number present in the dataset must be counted, allowing not only to understand the data dimension but also to limit the ML algorithms, since many of them require a minimum number of samples to achieve good results. Then, examining the number of network observations per label along with the mean, minimum, maximum, standard deviation and percentiles allows a concrete view of the dataset shape dataset and conclude on its balancing and completeness.

Another useful approach to gain insights about data nature comes from the plot histograms which shows the number of instances in relation to a range of values according to statistical measurements of specific features, as represented in the following Figure:


**Figure 3.7:** Representation in the form of histograms of some features obtained from network observations referring to a GRE flood attack. X-axis: observed values, Y-axis: count of occurrences.

From the analysis of the various histograms it is possible to infer about the activity time and therefore the periods of silence. For example, in a scenario with a highly intrusive bot the analysis of packets over time allows to infer about their activity since, contrary to the human actions that are usually paused, the actions of the less stealthy do not assume silence periods. In addition to this analysis, it is still possible to infer about sample scales and data distribution, which typically assume a preponderant position on one side of the histogram in relation to the median, which may hamper the patterns detection by ML algorithms.

### 3.3.2 Dataset correlations

Often, to calculate the relationship between the data is computed standard correlation coefficient, between every pair of features, that allows to uncover less intrusive patterns. Nevertheless, besides this coefficient it is also important to compute and plot the *scatter matrix* as illustrated in Figure 3.8, in order to evaluate about the positive or negative correlation between features. The diagonal from top left to right bottom would be full of straight lines(i.e., each feature directly correlates with itself), so instead it is plotted the representation of a histogram of each diagonal feature [129].

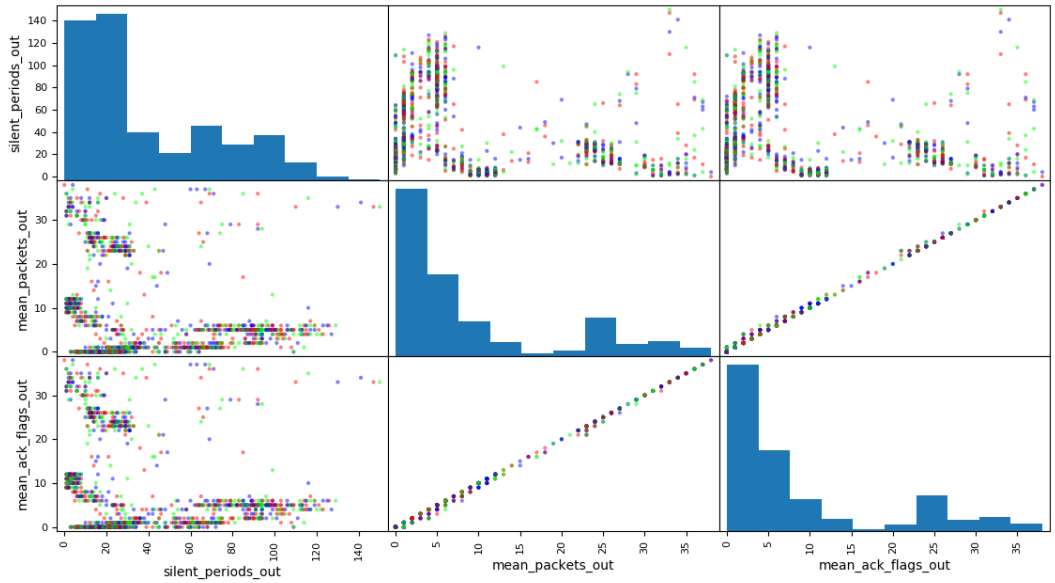



**Figure 3.8:** *Scatter matrix* for a ack flood dataset composed by three features. Each graph represents the linear correlation between each pair of features

From the graphs analyzes it is possible to conclude a few things. Firstly, there is a high correlation between the mean of outgoing packets and the mean of ack flags present also on outbound traffic, represented by the density of overlapped points. In this context, almost all of the outgoing packets carry an ack flag, assuming both features recurrently the same value, which can mean redundant variables. This is a simple example, however with bigger datasets the analyzes complexity growths exponentially, being impracticable the observation of all pairs of features. This *scatter matrix* contains 9 graphs for a dataset with only 3 features. For one with 30 would have to contemplate $30^2$ graphics, being therefore a non-scalable approach.

## 3.4 Data Pipeline and Knowledge Extraction

At this point, data is almost ready to be loaded to the data pipeline. Generally, and emphasizing that this work focuses on supervised learning, the network observations goes through a data pipeline, in which it assumes a maximum of 6 stages., as shown in Figure 3.9. However, this number is variable according to the algorithms, allowing to experiment and combine multiple data processing. Each stage is detailed in the following sections, according to the order defined in the Figure bellow.

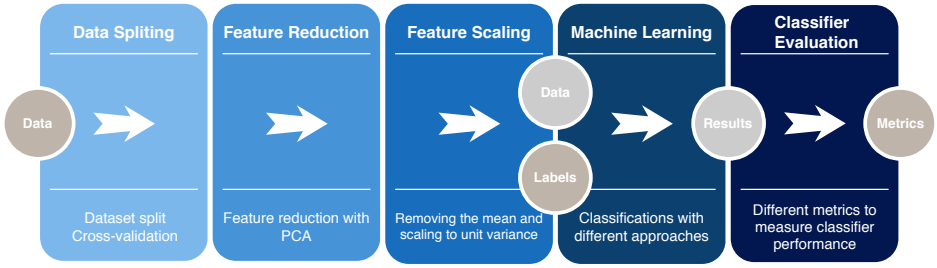



**Figure 3.9:** General representation of the data pipeline. The data are transformed along the pipeline to be suitable for each ML algorithm. Depending on the nature of these, some steps may be omitted.

### 3.4.1 Data splitting

Throughout the generic pipeline data division occurs twice, first at this stage, and then in the training phase of the ML algorithm. Regardless of the division point, it is necessary to guarantee the data representativeness of each subgroups in relation to the overall data. For the first division, which typically follows the follow the *Pareto* principle, two sets of data result: training set and test set, with ratios of 80 % and 20 % respectively. However, in volatile contexts where the samples number has a high variance and the consequent dataset size is small, it is necessary to ensure that the test set is data's representative, allowing to provide an unbiased evaluation of a model.

The cross-validation(i.e., the second data division) occurs in the model's training phase, resulting N training and validation sets, as illustrated in Figure 3.10. This strategy represents the crucial point where data plays a major role, and its representativeness determines the system's performance.

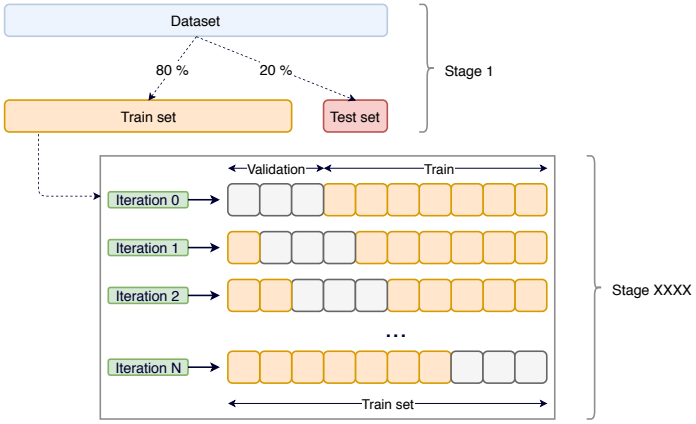



**Figure 3.10:** Data splitting applied on two distinct phases. Although split percentages are flexible, in the first split were defined by the *Pareto* principle.

Instead of shuffling the data and partitioning of the two sets in each iteration, leaving the representativity to the burden of luck, this process must be controlled in order to avoid biasing, and therefore, *stratified sampling* should be adopted. *Stratified sampling* ensures that the resultant train and validation sets maintain the same samples percentage over the

splits, and mainly that each validation set is overall data's representative. However, with the subgroups number's growth, it does not guarantee different subsets, and its repetition among the various splits may occur. Typically the splits number, called *folds*, is ten, often designated as magic number due to the excellent results it provides in general, resembling an axiom.

In addition to the first division, it is good practice to have one more set of data in order to evaluate the model in order to understand how anomalous behavior is detectable. For example, in a HTTP flood detection context, where normal activity is around 10 requests per minute, and in an attack it becomes 100, it is important to realize in this 90's failure how far it is possible to detect. Thus, this *zero-day* tests are required, varying in frequency and scales the various anomalous activities in order to provide a clear view on the generalization and detectability of highly masked patterns. A deep dive over this topic is present on the following Section 3.5.2

### 3.4.2 Features reduction

Feeding a ML algorithm with a set of features without being dissected is impractical, it increases the complexity to find a good solution reflected in time processing and not satisfactory results. This way, adjust feature dimensionality is mandatory on every ML problem. However, should be present the concern with relevant information, since with features reduction, some of them are discarded and the inherent information is lost. On the other hand, it filters noisy and poor-quality data along with unnecessary one, aiding to reach high performances. Another important vantage from features reduction is visual simplicity, once reducing the dimensions to one, two or three allows project the data in order to gain even more insights about activity patterns after the previous ones obtained from data's overview on Section 3.3.

The most notable dimensionality reduction algorithm is PCA. This technique projects the data onto a hyperplane preserving the maximum amount of variance, once it will most likely lose less information than other projections with lower variance. This hyperplane is previously identified by its approximation to the data.

Another important metric to is the mean square distance between the original dataset and its projection on a specific axis, which should minimize this value [129].

But, how should be chosen the right number of dimensions? The process should not be arbitrary and is important to choose a number that choosing the number of dimensions to reduce down to, it is choose the number of dimensions that add up to a sufficiently large portion of the variance(e.g., 95%). A descriptive approach is to plot the *explained variance* in relation to the features number, as portrayed in Figure 3.11. Typically, there is an elbow in the curve, where the variance ratio stops growing quickly.

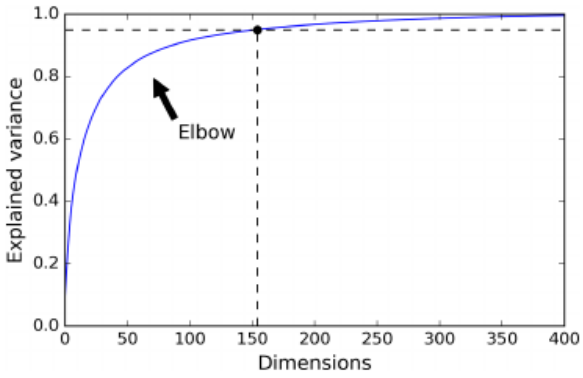



**Figure 3.11:** Representation of *explained variance* according to the number of dimensions [129].

In this case, reducing the dimensionality to about 150 dimensions would not lose too much *explained variance* and, therefore relevant information. It is possible to confirm that there are no valuable information lost by inverting the reduction process and verify against the original space that the data's correlation are preserved along with its variance, as shown in Figure 3.12.

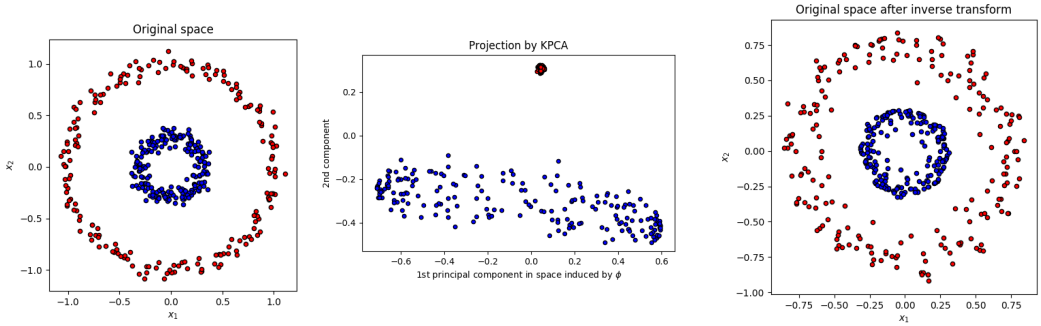



**Figure 3.12:** Demonstration of PCA reduction using Kernel PCA method, and inverse transformation from the resultant projection, maintaining all the relevant information and correlations [156].

As the data pipeline shows, this process is applied right after the split, on both train and test set. Despite the diverse advantages, the features reduction use is optional. However, consistency must be maintained, and once used at this stage, it will have to be used in subsequent tests, as in the *zero-day* test cases.

### 3.4.3 Features scaling

Often, a dataset is composed of several highly varied features not only in magnitude, but also in units and range, which may be a limitation, so these discrepancies have associated a greater weight to each feature, biasing the model. Thus, a relevant pre-processing technique, designated as feature scaling, must be applied to the data before being loaded to the ML algorithms.

Often, ML algorithms are invariant to the features scale, as will be detailed in Section 3.4.5. With a few exceptions, like SVM or NN, there are algorithms that present better performances

when the inputs numbers assume the same scale. Typically labeling each network observation is accomplished on forward stages, however in cases when are present on this one its values should not be submitted to a scaling process.

There are two main approaches to scale a dataset:

- *Normalization*

  Also known as *Min-max scaling*, this strategy is quite simple, each value is linearly transformed into a new value which end up ranging from 0 to 1. The value is subtracted by the min, and then divided by the max minus the min: $x \rightarrow y = (x - min)/(max - min)$.
- *Standardization* Differently, each value is subtracted by the mean(so standardized values have a zero mean), and then is divided by the variance, resulting a distribution with a unit variance. Unlike, the *standardization* result does not bound values to a specific range, being a constraint to some algorithms like NN that expect input numerical values ranging from 0 to 1. Nevertheless, this is much less affected by outliers. Comparatively to the previous strategy, and considering a dataset with outliers, the normalization scales normal data to a tiny interval, leaving the rest to a longer one, while *standardization* is not bounded and consequently does not lose important information.

As well as the previous, this process is optional albeit some ML algorithms benefits directly from its usage. Therefore, its coherence also must be maintained on both train and test sets, otherwise when applied solely on the train one, for example, when the algorithms faces test instances, from the test set or *zero-day* test cases, it will produce worst results due to scale variation.

### 3.4.4 Labeling approaches

Once the focus is on supervised learning, labeling each network observation is mandatory before forward it to the ML algorithms. There are multiple strategies to categorize each activity pattern. Considering an anomaly detection system, on a context where are three attack sources, each one with a different attack vector: the first one perform an UDP-based memcached, the second a DNS query flood and the third is exfiltrating data.

One prominent approach is to reduce to a *binary classifier*, where it only need to distinguish between two classes. Thus, all the three attacks are classified with the label one meaning that are all anomalies, being the label zero reserved for normal traffic. This is a basic approach, which is available widely on every classification algorithm. However, not always group different attack vector on a same groups is worth, once it may be useful to identify specifically which attack vector is present on a given moment in order to apply particular defense methods. Another limitation from the attack aggregation is the inherent difficult to generalize correctly for all three vectors, once each attack has its characteristics and dynamics that can bias the final classification.

Thus, since there are three different attack categories, a direct approach consists in label each network observation with the respective label, adopting a *multiclass* classification. Whereas *binary classifiers* distinguish between two classes, these ones can distinguish between more than two classes. Following this strategy each attack vector is assigned a singular

label, for example, label one for memcached, two for DNS query flood, label three for data exfiltration and finally the label zero, once again, is reserved for normal traffic. This way, the classifier is able to learn from the different observations, from distinct patterns and when face a new instance categorizes with a label from zero to three. At this point, the vectors segregation allows to apply different counter measures to each one, however, the general performance of the classifier may be affected by the similarity between attacks, introducing attack correlations that skew the classification. Besides this limitation, there are plenty of algorithms that does not support *multiclass*, which introduces the next approach that is a division of *multiclass* classifiers into sequential classifiers, where its number is defined by the number of labels. Thus, on this context, it would result three classifiers, where the first one identifies memcached attacks, the second DNS query flood attacks, the last one is responsible for identifying data exfiltration patterns and the last one manages normal network observations. Then, when a new instance arises, each classifier provides a decision score for the specific instance, and is selected the class whose classifier outputs the higher score, as illustrated in Figure 3.13. This is designated as *OvAs* strategy[4].

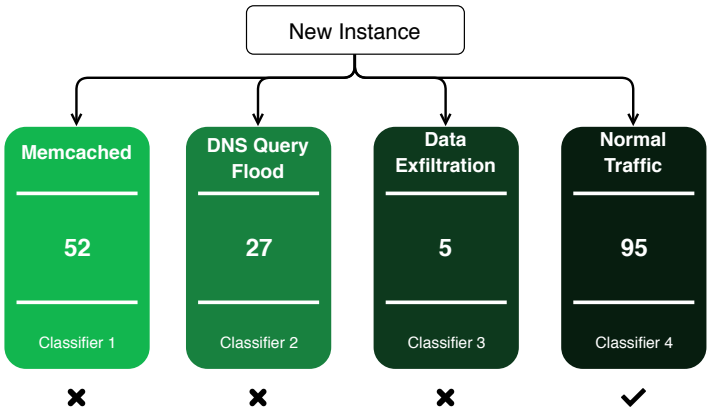



**Figure 3.13:** Representation of the OvA strategy, where the classifier responsible for normal traffic produces the highest score, being the new instance classified as normal.

Another possible strategy to combat the *multiclass* constraints, relies on the *divide-and-conquer* paradigm, dividing *multiclass* classifier into several *binary classifiers*. However, in this case each classifier needs to distinguish a par of attack vectors(i.e., between normal traffic and memcached attacks, between normal traffic and data exfiltration, between memcached attacks and data exfiltration, and so on.) This is called the *One-versus-One (OvO)* strategy. However, there is a major limitation, the number *binary classifier* grows rapidly with the increasing classes. For this example, it would result six different *binary classifiers*. Generalizing, for $N$ classes, it are necessary $N \times (N-1)/2$ classifiers. Thus, this approach is impracticable in most contexts since training only a singular classifier is computationally time-consuming, with the amount of data's addition and the necessity to train multiple classifiers it becomes extremely difficult to implement a *OvO* solution.

[4]Also known as *one-versus-the-rest* (OvR)

### 3.4.5 Outbound anomaly classification

As discussed in Section 3.2, the resultant features, try to express a network activity through its inherent patterns which should be inferred and segregated at this stage in order to classify news instances. At this point in the pipeline, regardless of the supervised learning algorithm used, the data is fully ready to be loaded.

#### 3.4.5.1 Support Vector Machine

SVM is a ML algorithm, where the main concept is to find boundaries between known samples, defining a segregation *hyperplane*. Based on these boundaries the algorithms decide how to classify a network observation, assigning to each observation the respective label. SVM is hypersensitive to features scaling, meaning that a data pipeline with a SVM in the final stages, must be preceded by a *standardization* stage. There are several SVM algorithms characterized by different parameters which should be tuned.

*LibLinear* is a linear SVM algorithm implementation, which fully determines a decision boundary based on the instances located at the edge, the *support vectors*. Often, this *hyperplane* formed by the decision boundary and respective margins(i.e., distance between decision boundary and respective *support vectors*) is designated as *street*. Ideally it should be as large as possible limiting also the *margin violations*(i.e., instances that end in the middle of the street or even in the wrong side) in order to maximize the classifier's perform [129]. This directive entitled *soft margin classification* is balanced by the $C$ parameter: it tells the algorithm how much to avoid misclassifying each training sample. For example, large $C$ values produce narrow streets, and all training samples tend to be classified correctly, avoiding violations. On the other hand, small values result in a *hyperplane* with wide margins, which is more conducive to missclassify. However, the tunning process, particularly on this penalty parameter $C$ requires a previous dataset overview in order to perceive a possible data's separation according to a *hyperplane*, as well as the number of outliers in the training samples. Figure 3.14 represents the $C$ parameter variation in relation to the results bias.

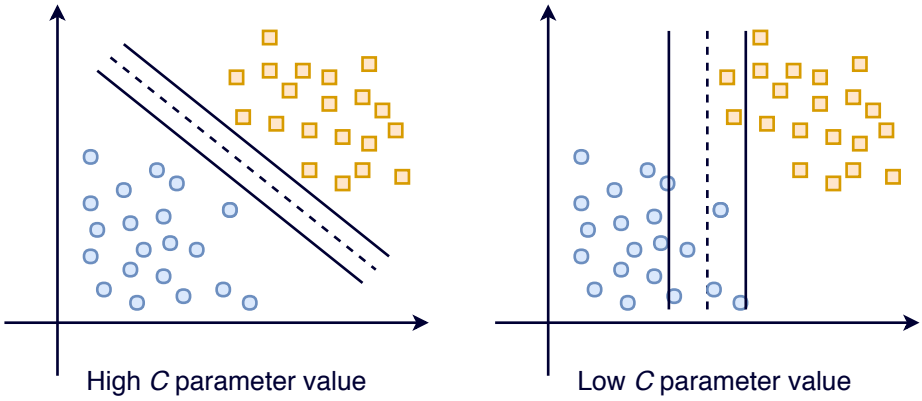



**Figure 3.14:** Representation of the impact of SVM's C parameter separation margin

Higher $C$ values mean less margin and therefore high penalty for misclassification. In its

turn, on the right side, small $C$ values, although many instances end up in the middle of the road, optimizes the classifier, which tends to generalize better, avoiding data *over-fitting*.

Note that linear SVM algorithm supports binary and multiple classes, and when training samples are labeled with more than two classes, it provides both strategies, OvA and OvO, detailed in the Section above 3.4.4. By implementing both strategies, it carries the advantages and disadvantages, emphasizing that the training process is extremely time consuming.

#### 3.4.5.2 Neural Networks

Generally, network behaviors and patterns are underlying excessive amounts of data, which reduce the SVM performance, thus representing a scalability problem. This limitation represents for the NN, a surplus value, since they benefit from the amount of data. Due to the constant growth of data and the application of ML methods in extremely complex areas and tasks, NN as core of the Deep Learning, has gained enormous popularity, much because of its versatility, scalability and power. Its application is in constant proliferation being actually present in contexts such as classifying billions of images(e.g., Google Photos and Google Images), speech recognition services(e.g., Apple's Siri), recommending millions of videos or music to users based on their history(e.g., Youtube and Spotify) and so on.

The NN structure can assume multiple implementations. Starting the simplest one, the *perceptron* which have multiple inputs weighted that tap into a single output. Multi-Layer Perceptrons (MLPs) implements a multi-layer structure, illustrated in Figure 3.15, combining various *perceptrons* and its internal organization, which may include *bias neurons* and *backpropagation* algorithms. This last one, works by analyzing output's error of the NN, the error is calculated by the input contribution of each neuron and its weight, and error minimization is achieved by varying the weights approaching the result as close as possible to the expected one. This process continues working its way backwards through the layers of the NN. The training is implemented using *Stochastic Gradient Descent* which try to optimize the weights according a specific solver [129].

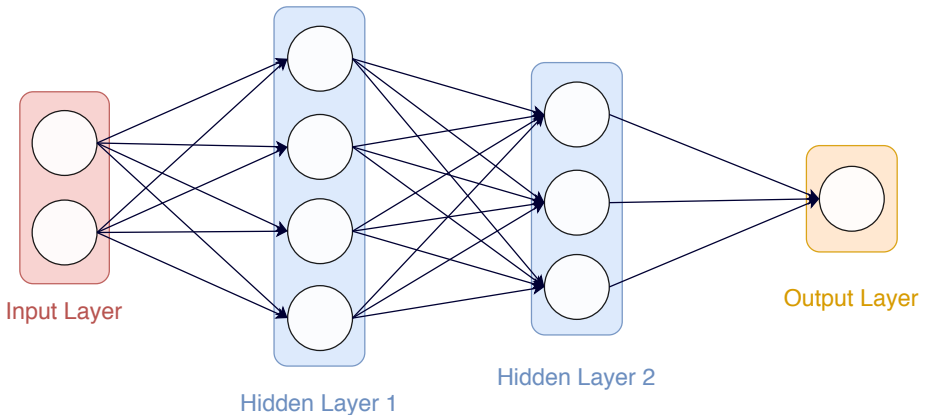



**Figure 3.15:** NN with an input layer, two hidden layers and a final output layer. When a NN has two or more hidden layers, it is called deep NN

Compared to SVM, the NN are also sensitive to feature scale, and therefore follow a data pipeline with equal preprocessing. Due to the plethora of possible layers, in which the size of each can vary, the hyperparameter tuning is too slow. In addition, since it is always the objective to maximize the classifier's performance, this process must be carried out exhaustively. Moreover, with a high number of layers, several free parameters appear between them that make the training process also slow.

#### 3.4.5.3 Decision Trees

Tree based learning algorithm are also a solution for supervised learning problems, widely present mainly due to its high accuracy, ease of interpretation, fading the concept of black box highly present in NN. Decision trees follow a hierarchical structure, as shown in Figure 3.16, consisting mainly of four elements:

- *root node*: the highest tree's point, it divides the samples into two or more homogeneous subgroups;
- *decision node*: apply a test function to an attribute, where the outcome defines the next branch;
- *branch*: determined by a subgroup of a entire tree;
- *leaf node*[5]: nodes do not split, forming localized regions where instances tend to fall, representing a class label.

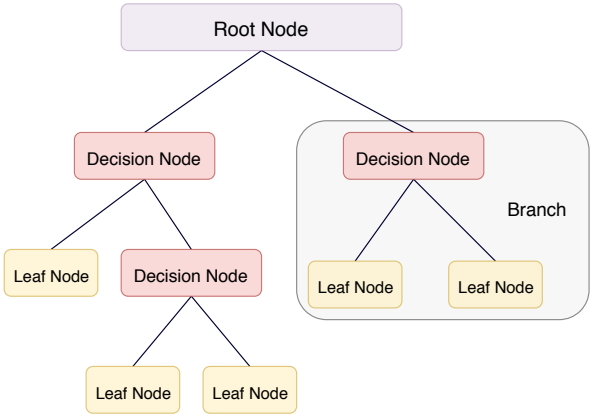



**Figure 3.16:** Decision tree structure example, with five *leaf nodes* and three *decision nodes*.

The tree's operation mode is based on the design paradigm divide and conquer whereby a *terminal node* is identified in a sequence of recursive splits in a typical small number of steps. Each step is mapped to a decision node, where it decides which branch to follow according to the outcome. This process starts at the root and recursively taper until hitting a leaf node, meaning a final classification. However, it is necessary to keep in mind the quality of each split, analyzing the *impurity measure* [157]. A split is pure when the route is only constituted by a single label, that is, for all the branches, all the instances choosing a branch belong to a single class. There are also other constraints that should be considered when tuning the predictor

[5] Also known as *terminal node*

parameters, for example, the tree's depth poses a major concern once when it assumes high values, the tree is able to learn very specific relationships incurring in *over-fitting*.

Due to its *white box* nature, these models are easily understood through its intuitive graphical representations which demystifies the underlying operations, highly abstracted in models such as NN. In addition to this advantage, decision trees are very useful for data exploration since it is possible to quickly identify relations between two or more variables allowing to gain more insights about the generalization process and the training set [158].

Comparatively to the NN, tree models require less data and are not influenced by outliers or missing values. In contrast, these predictors are invariant for monotonic transformations and therefore do not benefit from the feature scaling.

#### 3.4.5.4 Ensemble Methods

An *ensemble method* is an algorithm which combines a group of predictors(called *ensemble*) into a single one, maximizing its accuracy. Usually, these predictors are decision trees, which when grouped together constitute a more stable model that incorporates the inherent advantages of predictors and generally provides better predictions than an individual predictor. There are two main *ensemble methods*:

- *Bagging*: is a voting method whereby each predictor is assigned with a different data portion. The training set is divided along the predictors following a *bootstrapping* technique, as illustrated in Figure 3.17, where the instances are randomly splitted with replacement, in other words, it is possible for an instance to appear more often in various training sets than others, which in the worst case do not appear. Although each individual predictor has a higher bias than if it were trained with the original training set, the aggregation of several reduces not only the bias but also the results variance.

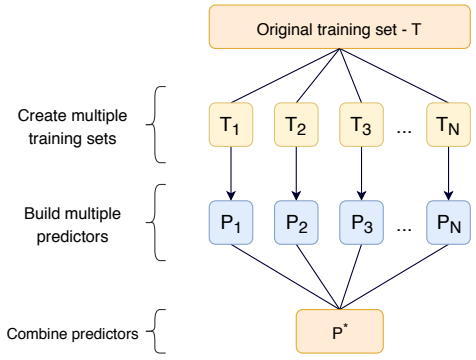



**Figure 3.17:** Representation of how the *bagging ensemble methods* operate on the training set, dividing it into several that feed the same number of predictors that ultimately unify, through a voting process.

The group of predictors when face a new instance, classify it individually, and through a voting process comes the final classification. The most popular implementation is the *random-forest*, which is based on a set of decision trees.

- *Boosting*: refers to a family of algorithms which combine several weak learners into a strong learner. The general concept of most *boosting* is to sequentially train predictors, each trying to correct its predecessor misclassifications, as represented in Figure 3.18. Conversely to the strategy described above, where the predictors act and parallel, in boosting methods the predictors assume a chain structure introducing dependency.

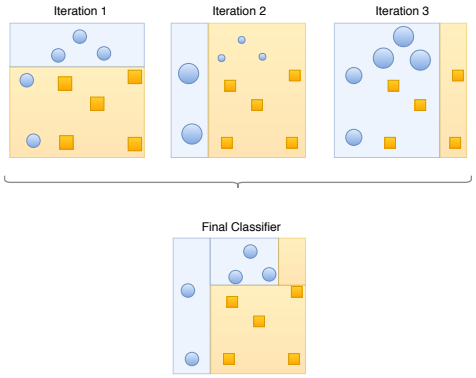



**Figure 3.18:** Representation of the *boosting ensemble methods* iterative process, intending to adjust the weight of an observation based on the last classification.

The most prominent algorithms that follow this method are *Adaboost*, *Gradient-boost*, and *Xgboost* that have recently been consistently used to gain machine learning competitions on Kaggle [159], [160].

Often these algorithms derive from decision trees which makes them invariant to features scaling, being the data pipeline's third stage omitted when empowered one of these *ensemble methods*. In addition, the process of tuning parameters becomes more complex, since there are the specific trees's parameters as the depth or maximum number of *leaf nodes*, and also the parameters of *bagging* or *boosting*. For instance, both *Adaboost* and *Gradientboost* need parameters to manage the *boosting* and model's robustness, such as the *learning rate* that determines the impact of each tree on the final outcome or the *estimators number* that represents the number of sequential trees to be modeled. Due to the complicity of these parameters, the variation of one must be reflected in the other proportionately, but in a different sense, specifically, the learning rate must be decreased, and consequently, the number of estimators must increase in the same proportion, thus obtaining a more robust model.

#### 3.4.5.5 Novelty and Outliers Detection

In an anomaly detection context, a straightforward approach involves the implementation of outliers detection. Both aim to separate a core of regular observations from the outliers, differing in the training process and consequent data. In novelty detection the training set is cleared of outliers, and these are detected in new instances. For once, in outlier detection, it is assumed that the dataset contains outliers which are contaminating it. Based on these assertions, the objective is to cope with its presence in the modeling phase of the normal pattern, adjusting one or more data cores, ignoring the divergent observations.

Usually, these detection methods are applied to datasets in which a huge number of normal observations are available and where there are insufficient data to describe abnormalities. Under these conditions the datasets is classified as unbalanced, once the examples number per class is highly different. These data sets are consistently present in complex industrial systems that aid in these methods to detect failures, as well as in electronic security systems, where the goal is to detect intrusions such as credit card or mobile phone fraud detection [161].

On both methods, a normality description is learned by constructing a model from the numerous positive instances(i.e., data that represent normal activities). Afterward, the model faces previous unseen patterns which are compared with the normality model providing a decision about its nature: inlier or outlier.

## 3.5 Knowledge Process Evaluation

### 3.5.1 Performance evaluation

Evaluating the performance of a classifier, regardless of its category, is often reduced to the analysis of accuracy, however this process should be taken more emphatically due to complexity and extensibility.

The first moment of evaluation should focus on cross-validation, detailed in Section 3.4.1. This method not only avoid biasing, but also elucidates the learning process, allowing to formalize an initial interpretation of the model. This perception comes from the analysis of multiple accuracies obtained at the end of each fold. In other words, the model in each iteration, trains with a training set's percentage and validates with the rest obtaining an accuracy per iteration, that when grouped demonstrate the model's behavior for the training data. The accuracy is a straightforward metric, calculated according to the following Equation 3.1.

$$Accuracy = \frac{Correctly\ classified\ instances}{Total\ instances} \tag{3.1}$$

From this process results the training accuracy, the average of all accuracies, which represents the first base term to conclude on the overfitting of the model. The second term is obtained at the end of the cross-validation, when the model faces the test set, where the analysis is more refined in order to obtain an exhaustive evaluation of the model. At this point, a new structure, called a *confusion matrix* and represented in Figure 3.19, is introduced, from which it is also possible to obtain the classifier's accuracy among other metrics.

| | Predicted Classes | |
|---|---|---|
| | **Positive** | **Negative** |
| **Positive** | **TP** - True Positive | **FN** - False Negative |
| **Negative** | **FP** - False Positive | **TN** - True Negative |

**Figure 3.19:** *Confusion matrix* for a binary classifier. The paradigm can be transposed to a multiclass classifier, where the matrix grows proportionally either in rows and columns, maintaining the correct classified instances on the diagonal from top left to bottom right.

Following an approach from the simplest *confusion matrix*, its first row concerns the actual positive class, that is, *true positive* when the samples are classified as positive, or *false negative* when misclassify, labeling as negative. In its turn, *false positive* is when negative samples are classified as positive ones, and *true negative* when negative samples are classified correctly, as negatives. For example, directly from the *confusion matrix* analysis, it is possible to calculate the classifier accuracy, the number of correct classifications being given by the sum of all diagonal values from the top left to bottom right(i.e., *true positive* and *true negative*), and the number total is given by the sum of all *confusion matrix* fields.

This is the base process for analyzing a *confusion matrix* of a binary classifier, however it is plausible to extend to a multiclass classifier with the nuance that analysis complexity increases due to the greater number of classes. Regardless of the labels number, these simple statistics present in the *confusion matrix*, are the foundations to calculate new concise metrics, which allow, for example, to corroborate the accuracies of one class over another in an unbalanced dataset, where the ruling class is more likely to have a higher precision. An interesting one to look at is the accuracy of the positive predictions, designated as the *precision* and defined according the following Equation 3.2:

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives} \tag{3.2}$$

However, evaluating a classifier based solely on two metrics is a reckless approach that does not translate the entire performance of the classifier. Another metric is often included is the *recall*[6], defined in Equation 3.3, which is the ratio of positive instances that are correctly detected by the classifier.

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives} \tag{3.3}$$

There is an inverse proportionality relationship between these two last metrics, called *precision/recall* tradeoff, that does not allow both to assume, simultaneously, high values. It is important to adjust this trade-off according to the context of the problem, finding the ideal point between the two extremes: low recall(i.e., some anomalies are not detected) and high precision(i.e., detects anomalies that are effectively minimizing false positives). Thus, in an

[6] Also known as *sensivity* or *True Positive Rate (TPR)*

anomaly detection context it is preferred to have high accuracy, tolerating some anomalies rather than having a high number of false positives.

Often, in order to find an optimal blend of precision and recall, both metrics are combined into a new one called $F_1$ *score*, defined by the following Equation 3.4:

$$F_1 = 2 * \frac{Recall * Precision}{Recall + Precision} = \frac{True\ Positives}{True\ Positives + \frac{False\ Negatives + False\ Positives}{2}} \tag{3.4}$$

The $F_1$ *score* is the harmonic mean of *precision* and *recall*, which punishes extreme values by giving more weight to low ones, whereas regular mean treats all values equally. For instance, a classifier with full *precision* and a *recall* of 0.0 has a regular mean of 0.5 but a $F_1$ *score* of 0, prevailing the lowest value. Thus, in order to create a balanced classification model, the $F_1$ *score* should be maximized, finding the optimal balance of *precision* and *recall*.

Beyond this trade-off, another popular designated as *Receiver Operating Characteristic (ROC) curve*, plots the *recall* against the *False Positive Rate (FPR)*. The FPR is the ratio of negative instances that are misclassified as positive, and can be obtained from the *true negative rate*[7]: $FPR = 1 - TNR$. Hence, the *ROC curve* plots *recall* versus 1− *specificity*.

Comparing multiple *ROC curves* against different ML algorithms provides a straightforward method to evaluate the classifiers performances on a anomalies detection context.

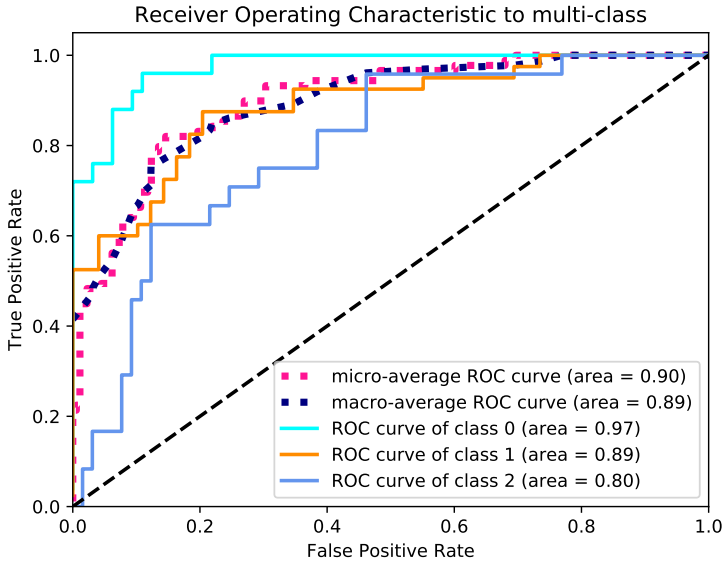



**Figure 3.20:** ROC curve for SVM multi-classification using Wine dataset [162].

As Figure 3.20 illustrates, the higher the *recall* represented by the Y axis, the more false positives the classifier produces. Ideally, good classifiers should be as far as possible from the dotted line, which is the *ROC curve* of a purely random classifier, trying to reach the

[7] Also known as *specificity*, which is the ratio of negative instance that are correctly classified as negative

top left corner, achieving the optimal point, where the FPR is zero and the TPR is one. Furthermore, another common way that allows comparing classifiers is measuring the Area under the curve (AUC). Whereas a purely random classifier is characterized by a ROC AUC of 0.5, a perfect classifier should have an ROC AUC equal to 1.

### 3.5.2 Zero-day tests

As the generic data pipeline in Figure 3.9 shows, this is the last stage and may even be omitted in simple scenarios. However, on anomaly detection contexts makes sense to corroborate the model's ability to detect. The processing of data on this stage must be exactly the same as those that fed the predictor, that is, from the acquisition process and subsequent operations, described in Section 3.2, to the stage of exclusive ML, maintaining the coherence so that the model can be evaluated correctly, partially simulating its application in a production environment. The exhaustive evaluation of the behavior and flexibility of the predictor when facing unseen instances is again performed with the help of the various metrics detailed in the section above. In turn, the granularity of the evaluation is directly related to the data present in the set of *zero-day* tests, which should preferably differ substantially from those used in the machine learning stage. This divergence can be emphasized in the frequency domain of the features, as well as in the scale, thus varying in a spectrum, where attack vectors belonging to one extreme are highly intrusive, while attacks closer to the other are stealthy, almost imperceptible.

Usually the attack vectors assume a unique and known attack dynamics, such as those discussed in Section 2.3.6. Thus the fluctuations along the spectrum are the result of the attack dynamic variation, differing mainly in the amplitude of the packet throughput and inherent bandwidth as well as the intervals of silence. For example, considering a HTTP flood scenario, where the classifier learned with a set of observations: web-browsing activity as normal data, and a intrusive HTTP flood as abnormal one, where every 1.5 seconds 10 POST requests were sent. At this point, it is important to note the flexibility of the classifier when it comes to classify samples that slightly or extremely differ from these dynamics. By facing lowering the intrusive degree it is possible to evaluate the classifier and establish a limit of detectability. In this way, it is possible to address a set of highly stealth *pulse-wave* dynamics that completely deviate from the 10 requests per 1.5 seconds, as shown in Table 3.3.

| Max packets | Silence duration | Accounted packets |
|---|---|---|
| 10 | 20 | 60 |
| 10 | 30 | 40 |
| 10 | 60 | 20 |
| 2 | 30 | 8 |
| 2 | 60 | 4 |

**Table 3.3:** *Pulse-wave* dynamics that vary the maximum number of packets as well as the periods of silences, which after observation for two minutes result in the total number of packets represented in the third column. For the sake of simplicity, it was assumed that the packet throughput is instantaneous.

Based on this approach, by facing the classifier with unseen dynamics(i.e., dynamics that were not present on the training phase, obtained from differing silences or mimicking on legitimate traffic, being similar to the normal human behavior) it is possible conclude, with more granularity, about the classifier performance.

In conclusion, following the premises discussed in the previous chapter, and given the diversity of business areas and the services that that today's corporations expose, the methodologies presented are transversal to all of them. The process must start with data acquisition, followed by the extraction of features and finally their transformation into knowledge. Since it is mainly focused on network anomaly detection and detecting unusual behaviors through network continuous monitoring and application of machine learning algorithms, there should be noted that different corporates have different profiles and distinct anomalies are described by different features.

CHAPTER 4

# Proof of Concept and Evaluation

*In this chapter is described the proposed solution to detect anomalous dynamics on a corporate network from its observation. The proof of concept is sustained by the procedure described in the previous Chapter, also following its order, from the generation of normal and abnormal traffic patterns using real attack vectors, composing the dataset, until the detection of anomalies according to each network observation, aided by algorithms machine learning and suggested metrics.*

## 4.1 Network Data Generation

Concerning the requirements described at Section 2.8 and 3.1, the implemented monitoring process includes a software-based solution probe belonging to the network domain, which passively captures raw packets on a wireless interface, storing all the data in a well-formatted document with *pcap* extension. The implemented sniffer requires read access to network adapter, correct system time and date, and some parameters related to the capture time, number of packets to be captured, or specific filters (e.g., TCP or UDP). Although all packet information is obtained, the processing of the metadata is strongly constrained between layer 1 and 4 of the OSI model. It should be noted that due to the passive nature of the sniffer, there is no perceptible impact at either the network level or the performance level of the system, thus avoiding implementation problems, often related to the deterioration of the overall performance due to the introduction of new features and services, which necessarily require specific resources to perform the tasks in full.

Typically an enterprise network follows a set of architectural principles [112](i.e., resiliency, redundancy, modularity and flexibility, ...) and a hierarchical topology composed of three layers, as shown in Figure 4.1.

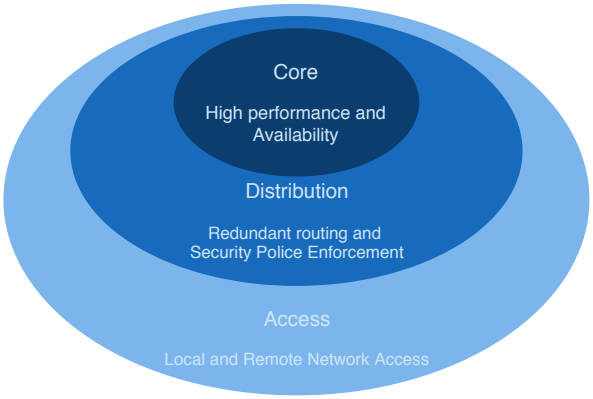



**Figure 4.1:** Enterprise hierarchical network layers

In order to build a baseline model to simulate, at a low level, a corporate network domain, the closest approach was obtained by using a home network, where the domestic private subnet represents all the company network and three layers were reduced into one symbolized by the home router. Due to hardware issues and in order to simplify the topology, despite the various devices connected to the home router, only one of them was monitored(i.e., Fedora 27).

There is a high contraction of network complexity compared to the level practiced in most of the organizations's topologies since all the layers have been aggregated and all the mandatory principles in physical scenarios do not matter in a simulated context because the network is simple and its control has been fully assured, and thus does not represent, any limitation to this Proof of Concept (POC).

From the establishment of this base network architecture, it is possible not only to determine the sniffer deploy position but also to define and simulate its behavior. Since the main purpose of this dissertation focuses on the detection of outbound attacks, in an enterprise environment, it is assumed that a device connected to the company network is already compromised, thus signifying the starting point for detection. Although the sniffer has been deployed as close to the host as possible, it can be easily transposed into higher layers, maintaining present the data representativeness and redundancy. In a real scenario, the ideal position would be between the layers of access and distribution, allowing the detection of egress attacks, in addition to enabling the use of traceback mechanisms or against more specific and effective measures. Thus, in order to generate network observations from which patterns will be inferred, and deviations between normality and abnormality, it is necessary to segregate business activities considered as normal from a set of attack vectors that produce abnormal network observations. These normal and abnormal behaviors fit into a classification problem allowing the use of ML models that follow the paradigm of supervised learning, as described in Section 3.4.5.

### 4.1.1 Normal Network Activity

The first step in resembling a corporate network begins with generating behaviors on the network that identify a set of activities, all of which are present in a company's everyday life. Thus, intending to replicate the normal habits of a common employee of a company, four practices often used in a corporate domain were selected [163], [164]:

- Youtube : reproducing videos at 720p;
- Spotify : reproducing random music;
- Twitch : watching a stream with source quality(i.e., between 480 and 720p60);
- Browsing : social networks and googling.

Obviously this list can be extended, there is a set of services inside the companies, many of them exposed in a TCP port(e.g., SSH on port 22, HTTP on port 80, etc.), however to keep the solution as generic as possible, were used services that are commonly available within the company and therefore are not blocked by the firewall. Although it is known that most corporations block the use of file-sharing services in peer-to-peer networks [165], such as torrent, thus closing the door attempts to enter the network, it has been purposely included in all normal services by reason that generates exaggerated amounts of traffic and allows the obfuscation of other services or anomalies.

Among the services presented, its network activity was registered for several periods of time, with a minimum of 5 minutes to a maximum of 15.

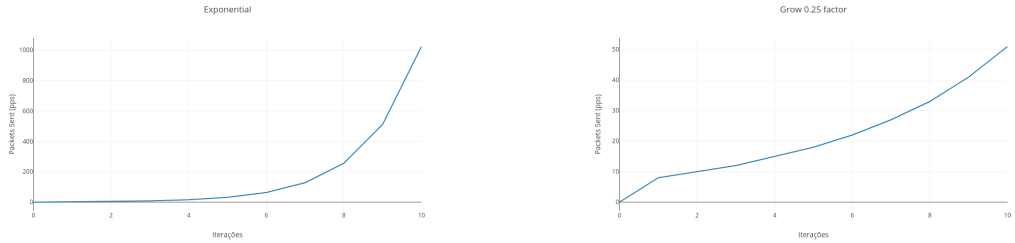
### 4.1.2 Abnormal Network Activity

On the other hand, in order to generate network observations which identifies anomalous patterns, it is necessary to create a set of attack vectors that define such behaviors. Consequently, being the *Mirai* botnet an icon in the DDoS history, and once its source code was made public on Github [166] for research and development purposes, was easily chosen as support to generate abnormal traffic. Besides, it implements ten attack vector, covering all three categories, which became famous by taking down many companies, as described in Section 2.4. Due to the extensive number of vectors, four were chosen based on the criteria that refers to frequency of use and bandwidth consumed:

- TCP SYN Flood - Protocol attack;
- TCP ACK Flood - Protocol attack;
- GRE Flood - Protocol attack;
- DNS Query Floods - Application-Layer attack.

Although *Mirai's* attack vectors often assume random ports, which may be blocked at the firewall, at the edge of the enterprise network, the most important features that are intended with the use of such vectors, is the similarity with reality and the ability to serve as the basis for generating abnormal dynamics, translated into abnormal network observations. By default, *Mirai's* attack vectors follow a constant flow rate, accentuating it intrusive character, as explained in Section 2.3.6. Initially, the traffic flood ramps up abruptly, reach a peak and maintain for a specific period of time. This sudden pattern immediately arouses the presence

of non-human behavior and as such allows one to more easily detect behaviors that follow these dynamics. In this way, the dynamics of action and pause of each attack vector were improved, obtaining vectors that follow *pulse-wave* as a possible way to perform the attack, thus distancing themselves in the initial highly intrusive nature, to more stealthy actions that resemble human behaviors, to a certain extent. In addition to this dynamics, three more were introduced, two of which are characterized mainly by their gradual growth and low throughput in the initial phases of the attack, as represented in the Figure 4.2.




**(a)** Exponential growth. **(b)** Linear growth.

**Figure 4.2:** Representation of both dynamics.

Both dynamics assume an iterative behavior, which is difficult to perceive if it is mapped in time and that despite the iterations being indexed in time, the mapping of the behavior in iterations highlights the increase of the packet throughput in relation to each iteration.

For the exponential dynamic, represented on the left, between each iteration there is a fixed silence period of 5 seconds, and the number of packets sent follows an exponential function with base 2. For the iteration $N$ are sent: $2^N$ packets. On the other hand, the dynamic on the right contemplates several periods of silence variable, inversely proportional to the number of packages sent. While the period of silence decreases by a factor of 0.25, the packet throughput increases by the same factor. For iteration $N$:

- Period of silence(PoS): $PoS_N = PoS_{N-1} - (PoS_{N-1} * 0.25)$
- Packets sent(PS): $PS_N = PS_{N-1} + (PS_{N-1} * 0.25)$

The last dynamic is intended to replicate the human behavior, as realistic as possible, by varying the periods of silence and number of packets sent according to an exponential distribution, thus generating a profile of abnormal activity that is highly reliable. In order to generate the distribution, the *exponential quantile function* was used, defined by the following equation:

$$F^{-1}(p;\lambda) = \frac{-ln(1-p)}{\lambda},\ where\ p\ is\ a\ random\ 0 \leq p < 1 \tag{4.1}$$

The $\lambda$ is the scale parameter which was chosen by following a discrete distribution in order to control the spread of the resultant exponential distribution. This distribution was used solely to obtain values for the duration of the periods of silence, deciding to keep the packet rate constant so as not to introduce too much abstraction and randomness in the observations of the network, allowing the awareness and understanding of them in a spectrum of intrusiveness of the attack.

Table 4.1 summarizes the previous detailed and implemented dynamics, suggesting a nomenclature for each one.

**Table 4.1:** Dynamics used during the acquisition of attack data, the first being more intrusive and completely immutable, while the others assume some parameters in order to vary the degree of intrusiveness.

| Dynamic | Description |
|---|---|
| DYN-CR | Constant rate - default |
| DYN-PW | Pulse-wave |
| DYN-EG | Exponential growth |
| DYN-LG | Linear growth |
| DYN-ED | Exponential distribuition |

Additionally to previous described attack vectors, the UDP-based memcached vector detailed in Section 2.3.5, which was used against Github recently and has the highest recorded peak, was also added, culminating the deficit in the volume-based category. By synthesizing and facilitating subsequent references, the nomenclature is established for each attack vector, as shown in Table 4.2.

| Attack Vector(AV) | Description |
|---|---|
| AV-1 | TCP SYN flood |
| AV-2 | TCP ACK flood |
| AV-3 | GRE flood |
| AV-4 | DNS query flood |
| AV-5 | UDP-based memcached |

**Table 4.2:** Pool of attack vectors and respective alias

In this way, five notable attack vectors were assembled that allow to shorten the difference between a real and a simulated scenario, creating attack profiles as close as possible to those that occurred in the recent past. It should also be noted that, the effectiveness of the attack vector is an irrelevant factor, the important thing to retain is the vector fingerprint and its dynamics, making the period of activity and silence vary, both in scale and frequency.

In addition, it is equally important to note that, regardless the attack vector, its action was captured either exclusively or simultaneously with the various services described in the preceding section, masking the attack with normal traffic. Despite the presence of other services, intrusive dynamics were tendentiously chosen.

With respect to the *zero-day* tests, the same process was adopted generating an extensive set of captures that includes the variation of the dynamics of each attack vector on the intrusiveness spectrum, particularly focusing on the stealth extreme. Despite the preference for less perceptible actions, the simultaneity of the services considered was also introduced. In this way it is possible to comprehensively analyze the behavior and flexibility of the system for diverse scenarios, perceiving both the generalization degree and the detectability in contexts with the presence of smarter attack vectors, that strongly resemble human actions.

## 4.2 Parsing packets to metadata

After data acquisition, it is necessary to start the iteration process on the network packets available in the various *PCAP* files, retaining only metadata containing significant information. These metadata are the result of the filtering process, as described in Section 3.1, where the proposed solution only contemplated two main transport protocols (i.e.,UDP and TCP), and the information produced is equal to the fourth exclusive layer of the OSI model:

- Layer 1 to 3:
  - source and destination IP addresses;
  - volume of bytes carried;
  - arrival Unix epoch time.
- Layer 4:
  - TCP:
    - source and destination ports;
    - sequence of flags: SYN, ACK, FIN, PSH, RST.
  - UDP:
    - source and destination ports.

Since this process is the initial phase of the solution, it must be done considering the overall performance of the system since, given the growth of the number of packets circulating in a corporate network, a bottleneck in preliminary phases questions the response time of the system. Thus, in order to achieve faster processing, scalability and ambition for a near-real-time system, this process was carried out with the help of a library, designated as *kaitai struct*, whose objective is to allow a new way of developing parsers for binary structures, such as the structure present in the *PCAP* files [167]. According to the benchmarks [168], Python version 3.5.1, it can parse 31.925 packets per second, whereas competing libraries are set at a level below 5 000 per second. However, these values can increase considerably when using another programming language, such as Java where the number of packets processed per second rises to the order of 121,000.

In addition to this boost in the solution's performance, it is also necessary to know *à priori* the capture time (i.e., the timestamp of the last packet minus the first). A straightforward solution would be to load all the packets into memory and then get the first and the last and start the processing, however this process, besides being computationally time-consuming, requires a lot of resources. Another solutions would be to iterate twice over the packets, in the first would be solely to extract the total capture time. Thus, in order to reduce the number of iterations to one, when capturing the raw data packets, a feature has been added to the sniffer that appends to each *PCAP* file a configuration file with the start time of end of capture, for later reading and indexing samples as described in the Section 4.3.

Since the nature of each observation is known *à priori*, as shows Table 4.3, 6 different datasets were created, with the aim of applying only binary classifiers.

| Datasets | Data obtained from |
|---|---|
| DT-0 | normal + all attack vectors |
| DT-1 | normal activities + *AV-1* |
| DT-2 | normal activities + *AV-2* |
| DT-3 | normal activities + *AV-3* |
| DT-4 | normal activities + *AV-4* |
| DT-5 | normal activities + *AV-5* |

**Table 4.3:** Summarizing the contents of each dataset, considering the various dynamics and simultaneity with normal services, as described in Section 4.1.2. The presence of network observation from normal activities is common in all datasets.

Initially, it was only considered a single dataset(i.e,g, identified as *DT-0*) with all samples belonging to all network activities, properly identified with *binary* labels, where label 0 represents all the normal network activities whereas label 1 identify anomalous behaviors. However, the aggregation of all network observations in a single dataset could increase the difficulty of pattern detection and hence the accuracy of the detection. Although all the vectors of attack represent anomalous actions in the network, each one translates a different pattern / fingerprint, which as a whole may preclude the correct generalization of a model. Thus the dataset was segmented into five distinct ones, each of which consists of normal and data from a single attack vector. It combines with the *OvA* strategy, as discussed in Section 3.4.4, where each dataset is assigned to a different ML algorithm, also a *binary classifier*, responsible solely for the detection of a single attack vector and inherent dynamics. For the five news datasets, the normal data is identified by label 0 while those belonging to each attack vector are recognized by the label 1.

## 4.3 Feature Extraction

Despite the definition of each dataset, in this process the metadata remains raw does not represent any kind of information, being necessary the advance to the chain's second stage, where a *statistical processing* is applied over the previously obtained metadata resulting the attributes, present on the Table 4.4, which constitutes the body of the sampling windows, being the first step to model the inherent patterns.

| Attributes | Description |
|---|---|
| packets_out | Number of packets transmitted |
| packets_in | Number of packets received |
| priv_src_ports_out | Number of privileged source ports present on egress traffic |
| src_ports_out | Number of non-privileged source ports present on egress traffic |
| priv_dst_ports_out | Number of privileged destination ports present on egress traffic |
| dst_ports_out | Number of non-privileged destination ports present on egress traffic |
| syn_flags_out | Number of packets with SYN flag set on egress traffic |
| ack_flags_out | Number of packets with ACK flag set on egress traffic |
| fin_flags_out | Number of packets with FIN flag set on egress traffic |
| psh_flags_out | Number of packets with PSH flag set on egress traffic |
| rst_flags_out | Number of packets with RST flag set on egress traffic |
| upload_bytes | Total number of uploaded bytes |
| download_bytes | Total number of downloaded bytes |
| connected_ips | Number of different IPs contacted |

**Table 4.4:** Low level description of the attributes present on every sampling window.

Initially, both strategies were considered, controlling only the outgoing traffic or in both directions, resulting in a number of attributes close to double the previous ones. However, by comparing the two models, many attributes related to ingress traffic did not contribute positively to the generalization of the model. On the contrary, it was introduced a lot of noise, resulting less satisfactory, and therefore only 2 attributes were constituted referring to traffic in this direction: *packets_in* and *download_bytes*.

In this way, for each packet is analyzed its direction by comparing its source and destination IP address with the internal corporate simulated network, being defined as external communication when the source belongs to the private network and the destination IP does not. Regarding the *AV-5* where the source IP address does not belong to the considered network, once it applies the ip spoofing technique, it is treated as egress traffic. No internal communications were contemplated, since they do not fall within the scope of the solution. Nevertheless, both egress and ingress traffic are analyzed, resulting several sampling windows, which contains the previously listed attributes sustained below:

- Packets transmitted

  The number of packets transmitted allows the distinguish between periodic automated communications and the analyzis of silence periods.
- Packets received

  The number of packets received allows to infer about the trend of communications within the company, that is, it serves as a comparison term with the previous attribute to analyze the ratio between packets sent and received, and to perceive the tendency of the traffic direction.
- Privileged and non-privileged ports

  Typically the services provided by the enterprises are exposed on privileged ports(e.g., web server on port 80 and 443) and the inherent data is generated from these ports. On the other hand, many attack vectors use unprivileged ports at random, as these do not

require administrator privileges to perform communications, allowing network malicious activities to be performed.

- TCP flags

  The ratio between TCP flags may highlight unusual network traffic, in addition to the unbalance counting of these is a strong indicator of malicious activity on the network. For instance, under a ACK flood, independently of its intrusive degree, it is expected that the count of ACK flags, is higher than the rest.

- Upload and download bytes

  Usually endpoints generate more download than upload, that is mapped on the download/upload ratio. For example, high intrusive DNS query floods are reflected into higher upload total bytes, thus representing a change on the normal pattern.

- Connected IPs

  Often corporate devices tend to communicate consecutively to the same IPs addresses, resulting a low variance, whereas a compromised device tend to communicate with a pool of unusual IPSs, therefore, increasing the variance.

It is important to point out that no specific attributes were adopted for each attack vector since, in addition to introducing a limitation on the scalability of the solution that would be confined to a small group of attack vectors, it also flees somewhat of the scope once that these vectors were used in order to generate anomalous network dynamics, and not serve as a baseline to detect only this group of attack vectors.

From the second stage of the network observation generation chain result multiple sampling windows, each one containing 14 attributes. With regard to the size of each sampling window, and taking into account the assumptions reported in Section 3.2.1, a small scale of only 0.1 seconds was adopted that, although each window contains few data, allows the detection of periodic events, in addition to contribute to the rapid response of the system.

However, these set of windows need to be transformed into relevant network observations that effectively represent a pattern of a network activity. Thus, as described in Section 3.2.2, the sliding windows paradigm was applied to the sampling windows, in which the *mean*, *median* and *variance* were calculated on each column, resulting in a set of 42 time-independent features (i.e., 14 attributes * 3 statistical operations) that characterize each network observation. In addition, the number of silent periods, represented by the sampling windows where the number of packets belonging to outbound traffic is null, as well as the average duration of these periods, prefers a total of 44 features. Both features are closely related to the periodicity of traffic allowing to infer about human or non-human behavior, since, typically, autonomous actions are periodic in time, maintaining strict silence periods, or in case of anomalous network activities highly intrusive, do not contemplate any period of silence. In this way, for each observation window, the features distribution is presented in the Figure 4.3.

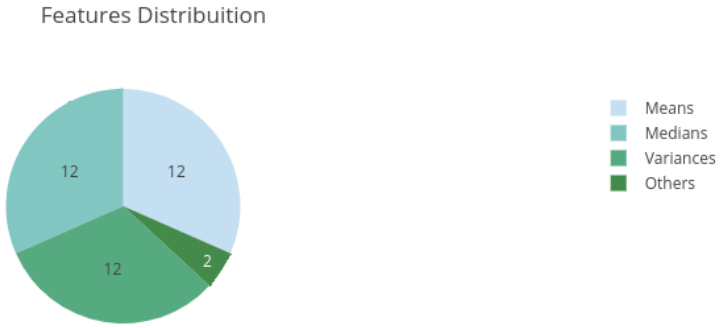



**Figure 4.3:** The three equal parts represents the 12 means, medians and variances that result from the application of each statistical operation to each attribute. The last two relate to the number of periods of silence and their average duration. Thus, each observation window contemplates this set of 44 features.

As in the sampling windows it is important to define the size of each sliding window in addition to the shift time, as described in Section 3.2.2. In order to obtain a nearer real-time response time without compromising the system's detectability, adopted a size of 1 minute for each sliding window with a shift time of 10 seconds. Thus, each sliding window consists of a set of 600 sampling windows (i.e., sampling windows size/sliding window size = 0.1/60), in which between consecutive iterations, 500 prevail, with only 100 being added (i.e., shift time/sampling windows size = 0.1/10).

The data is ready to go to the data pipeline, as detailed in Section 3.4, responsible for diverse operations, including applying ML algorithms to infer patterns, create a profile and identify deviations on these, allowing to detect malicious network activities.

## 4.4 Dataset Overview

Before proceeding to the data pipeline it is important to analyze the data at a high level, as demonstrated in Section 3.3, allowing to gain multiple perceptions, among them the data shape. Thus, from the various network captures, 4.685.935 packets were obtained after filtering by transport protocol (i.e., TCP or UDP), for a total of 470 minutes, distributed by the five vectors of attack and normal activities as shown in Figure 4.4.

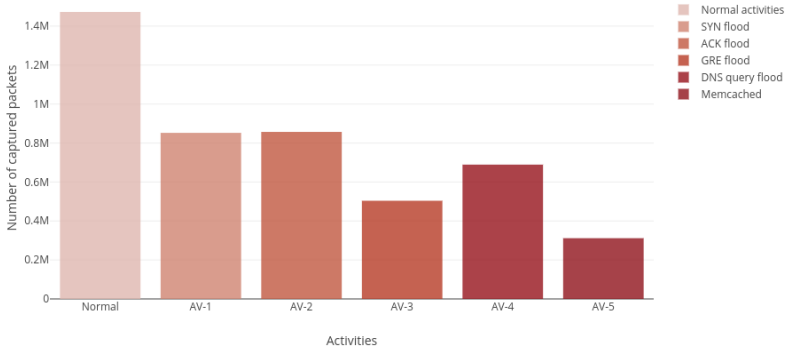



**Figure 4.4:** Number of captured packets per attack vector including also normal network activities.

The number of packets captured is strongly linked not only to the capture time and the underlying services, but also to the dynamics of an attack vector. For example, considering the action of two attack vectors, captured during the same period of time, without the presence of other services, it is linear that the vector with a more intrusive behavior will debit more packets than the stealth one (e.g., the number of packets resulting from a *DYN-CR* dynamic is larger than a vector that assumes a *DYN-PW* dynamic). The same occurs for different services, because two attack vectors with the same dynamics, captured during the same period of time in the presence of two very different services (e.g., torrent and spotify), it is normal that the number of packets obtained from the simultaneity of any vector of attack with the torrent is eminently larger.

For the first 4 attack vectors, the various network captures resulted in duration of 55 minutes each, including not only the combination of the various dynamics but also the presence of several services that define the set of normal network activities and that mix the normal traffic with that of attack. Despite equal capture times, the variation of services, dynamics, and attack vectors result in different packet numbers, as illustrates the Figure 4.4. Keeping the same capture time, the memcached attack vector was safeguarded from these variations, and its action was captured exclusively so that it was possible to obtain a model in which the normal and abnormal data were widely different, thus evaluating the degree of generalization as well as acting as a basis for comparison with the other models.

From these captures and inherent packets, were produced 2545 network observations which reflect the patterns of the various activities in the network, in observation windows of 1 minute containing a set of 44 features. The distribution of the various network observations is shown in the Figure 4.5.

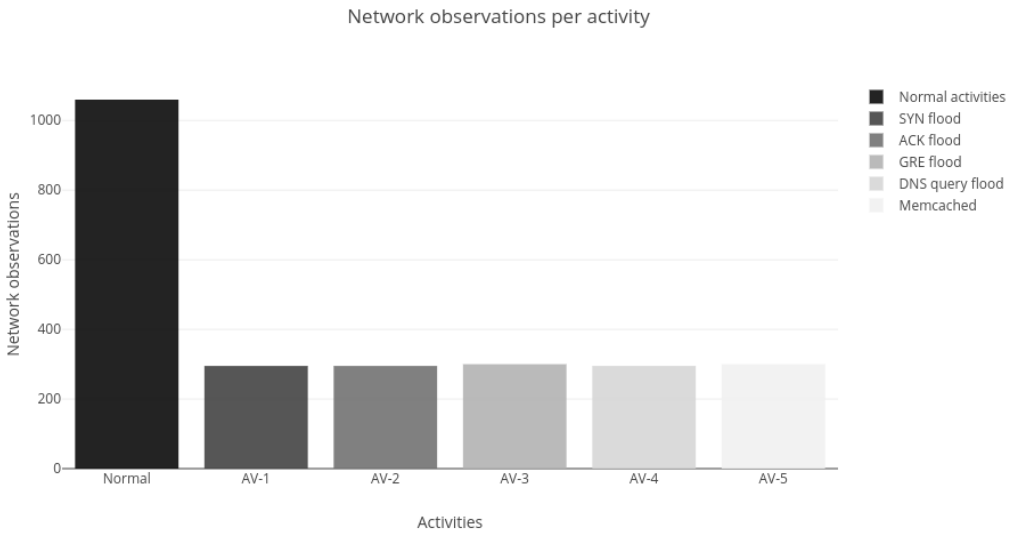



**Figure 4.5:** Number of generated network observations per attack vector, including also normal network activities

These sets of observations compose the various datasets, for example, *DT-0* is the aggregation of all 2545 observations, while the remaining five are made up of observations of normal activities in the network and of a single attack vector. The huge discrepancy, represented in the previous Figure 4.5, between network observation windows on normal and anomalous network activities(e.g., in the presence of one of the attack vectors) translated into the body of the datasets *DT-1* to *DT-5*, where the percentage of normal network observations significantly overlaps the anomalies, with around 78% against 22%, allow to characterize the five datasets as unbalanced. This characteristic may represent a limitation on the number of network observations required for a given model to generalize correctly, which will be further supported in Section 4.5. On the other hand, in relation to the dataset *DT-0*, the sum of the network observations of the various attacks ranging from 295 to 300 per each prefers a total of 1485 observation windows that when compared to the 1060 of normal activities allow to define the dataset as balanced.

Regardless of datasets, it is fundamental to gain insight into the patterns inherent in a set of network observation windows, a deeper analysis is needed of the various sliding windows as well as the features that define each one. Table 4.5 shows a view of various network observations from the *DT-1* to *DT-5*.

| id | network activity | silent_periods | mean_packets | ... | mean_syn | mean_ack | ... |
|---|---|---|---|---|---|---|---|
| 200 | Normal | 99.00 | 3.00 | ... | 0.00 | 3.00 | ... |
| 1071 | AV-2 | 12.00 | 25.00 | ... | 0.00 | 25.00 | ... |
| 1344 | AV-1 | 4.00 | 39.00 | ... | 39.00 | 0.00 | ... |
| 1354 | AV-1 | 2.00 | 2.00 | ... | 2.00 | 0.00 | ... |
| 1090 | AV-3 | 10.00 | 25.00 | ... | 0.00 | 25.00 | ... |
| 1100 | AV-2 | 4.00 | 0.00 | ... | 0.00 | 0.00 | ... |
| 1351 | AV-3 | 31.00 | 23.00 | ... | 0.00 | 23.00 | ... |
| 1210 | AV-4 | 21.00 | 18.00 | ... | 0.00 | 0.00 | ... |
| 1211 | AV-4 | 18.00 | 18.00 | ... | 0.00 | 0.00 | ... |
| 1215 | AV-4 | 8.00 | 18.00 | ... | 0.00 | 0.00 | ... |
| 1266 | AV-5 | 0.00 | 1.00 | ... | 0.00 | 0.00 | ... |
| 636 | Normal | 2.00 | 9.00 | ... | 0.00 | 9.00 | ... |

**Table 4.5:** View of a random portion of samples that make up the total dataset. Each column represents a feature while each row is a sample. Should be noted that the first column is a simple identifier, and all the features present in the table relate to outgoing traffic.

In the absence of knowledge about the underlying services, either for normal traffic data or for abnormal traffic, including the abstraction of the attack dynamics, besides the perception of the form of the data, it is possible to collect relevant information from its analysis and comparison with others samples. Given the *à priori* knowledge about each attack vector, the collected information must thus converge in an expected pattern. For example, in a *AV-1*(i.e., sample 1344) it is likely that the average of SYN flags will outperform all others. Likewise, it is also assumed that the use of normal services has more periods of silence than any other automated malicious activity. From the analysis of the table 4.5, it is still possible to think of a correlation between two or more features. Although, without the help of the *scatter matrix*, as detailed in Section 2, by looking only to the packets count mean and the mean of SYN flags for the *AV-1* and ACK flags for the *AV-2*, it is possible to verify that both are related because they assume the same value. For the *AV-1* case(i.e., sample 1344 and 1354), both the packet mean and the mean of SYN flags are equal, that is, on average each packet leaving the network contains active the SYN flag, which may indicate the presence of a *DYN-CR* attack dynamic.

Looking deeper, for sample 1100 in specific, it is interesting to see how during the time period of a minute four periods of silence were recorded and on average 0 packets leaved the network. Certainly, these periods of silence were relatively long and interrupted by sequences of few packets, however, given the depth of the analysis, the represented small features number and the inability to correlate with other samples, always considering that this is the result of a sliding window process, all information inferred visually about the possible dynamics of attack, or co-presence with a specific service is merely speculative.

As mentioned in Section 3.3.1, another way to analyze the datasets concerns representation in the form of a histogram, mapping features to their occurrence. Similar to Figure 3.7, the histograms in Figure 4.6 show the distribution of some features that reflect the exclusive behavior of an intrusive ACK flood attack, which follow a *DYN-PW* with an amplitude of

one hundred packets, between periods of silence of ten seconds.

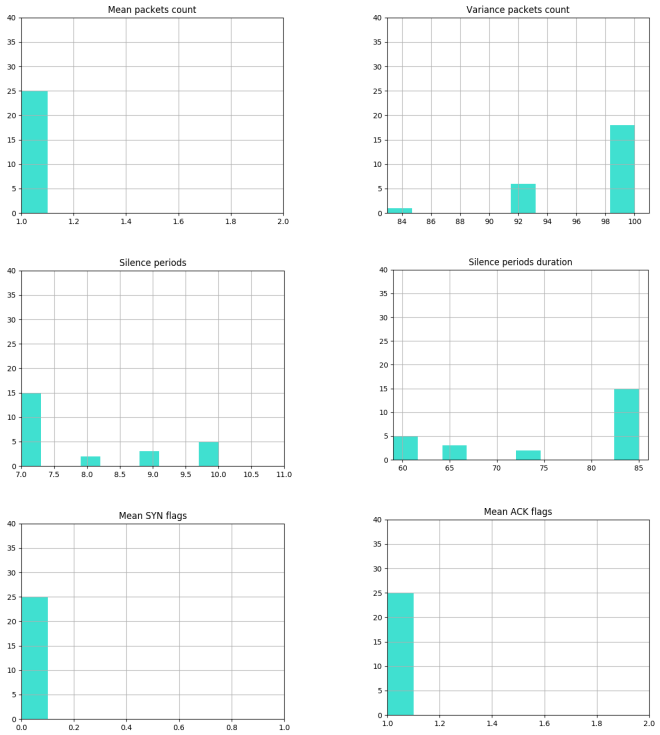



**Figure 4.6:** Histograms of some features present on observations obtained from an ACK flood scenario.

Due to the equality between the histogram that represents the average of packets and that which represents the average of ACK flags, corroborated with the little dispersion existing in the periods of silences, it means that there are short silence periods (i.e., predominant silence periods duration(in sampling windows): 85 * 0.1 = 8,5 seconds), which matches the 10 seconds. In addition, it is also correct to mention that each outgoing packet contains the ACK flag active, also matching the pattern of an ACK flood. This attack with such an intrusive dynamic aggravated by the absence of the presence of legitimate traffic related to other services becomes a simple case, where the correlations are explicitly and easily detected with the naked eye, whereas in more stealthy dynamics, the recognition of such subtle patterns can only be effective with the aid of ML algorithms.

In addition, due to the segregation of the datasets, it is important to gain insights on each one by analyzing the PCA, intending to reduce the number of features of each one, as discussed in Section 3.4.2. Starting with the larger dataset, which aggregates all network observations, Figure 4.7 plots the variance ratio as a function to the number of feature.

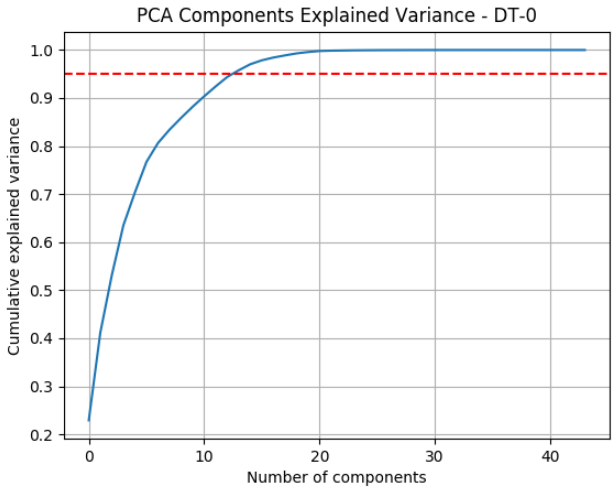



**Figure 4.7:** Representation of *explained variance*, the smaller the number of features the smaller the variance ratio and therefore the greater the difference between what the pattern expresses in relation to the original.

According to the graph, the ideal number of feature for *explained variance* ratio of 95% is 12 features. Thus, 95% of the dataset variance prevails, reducing the number of features to about one-third of the original 44 features. This possibly means that between these 12 the correlation is high, being able to be observed through the *scatter matrix* as demonstrated in Figure 3.8. However at this level and given the high number of features that would result in a set of $12 * 12$ graphs, it becomes impossible to analyze such correlations.

As for the other five datasets, the same analysis was performed from the graphs aggregated in Figure 4.8.

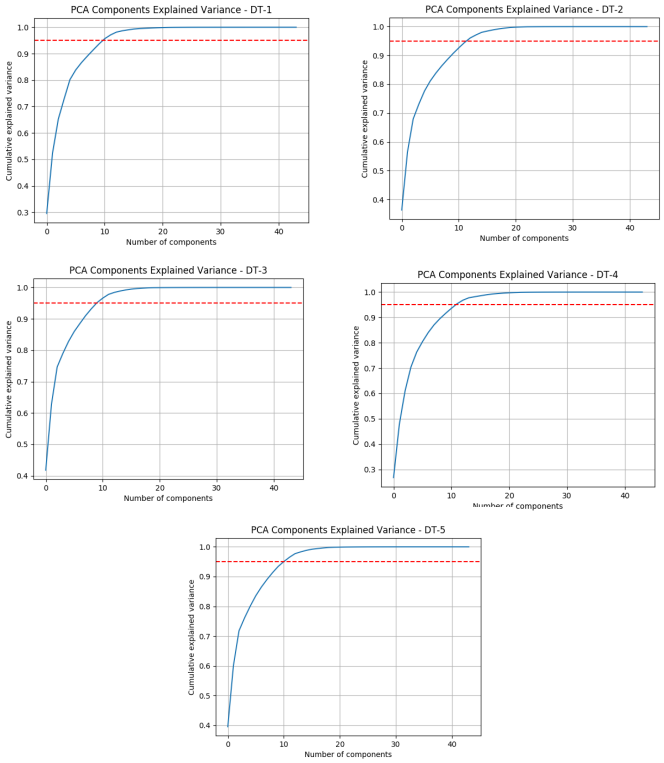



**Figure 4.8:** Representation of *explained variance* in function to the number of features, for each dataset.

According to the analysis of both Figures 4.7 and 4.8, is kept about 95 percent of the variance of each dataset with about 10 features. This reduction allows not only the decrease in the number of dimensions of each dataset and inherent complexity, but also contributes to reduce the processing time for the generalization of a model, as mentioned in Section 3.4.2. Finalizing the analysis of the various iterations of PCA reduction, Table 4.6 is formalized based on the previous figures, summarizing the optimal number of features that should be used for each dataset, without compromise the overall solution performance.

**Table 4.6:** Number of components to kept for each dataset.

| Datasets | Features number |
|---|---|
| DT-0 | 12 |
| DT-1 | 9 |
| DT-2 | 11 |
| DT-3 | 8 |
| DT-4 | 10 |
| DT-5 | 9 |

## 4.5 Classification Methods and Performance

Hereupon, each dataset was entered into a specific data pipeline, defined mainly by the penultimate stage, where effectively the various ML algorithms are applied, which in turn

directly influence the previous phases, as discussed in Section 3.4. However, there are stages and small processes that are transversal to all data pipelines, regardless of the classifier. The first phase (e.g., data splitting) is common in all pipelines, where initially the data are divided into two subsets, the training and the test, following the percentages defined by the *Pareto* principle, ensuring the representativeness of the data. This is an important requirement which has been kept in mind in applying the *cross-validation* process, by adopting the *stratified sampling* method, as mentioned in Section 3.4.1. This process also prevails in all pipelines, ensuring the same percentage of data for the various subsets of training and validation over the ten folds considered. Having said this, it is important to realize that from each data set one or more models are created to classify the test data of the respective dataset and other unknown traffic.

### 4.5.1 Neural Networks

In the ML stage, the NN were applied as the first algorithm. Since it is sensitive to feature scale, the feature scaling phase was introduced in all data pipelines just after the feature reduction stage. Regarding the algorithm, in order to maximize the outcome, the hyper parameters were obtained for each dataset, taking into account the optimal number of features for each one. An important aspect is the simplicity of the models, which independently of the datasets, are composed of a single layer, being designated as: *single-layer NN*. As mentioned in Section 3.4.5.2, typically neural networks must be associated with large amounts of data, and since the six datasets are not extensive, deep neural networks were not included in the machine learning phase of the various pipelines.

Table 4.7 summarizes the accuracies of six simple *binary* models after the ten splits on the *cross-validation* processes.

| Model | Dataset | PCA | Accuracy |
|---|---|---|---|
| ANN0 | DT-0 | 12 | $0.985 \pm 0.0146$ |
| ANN1 | DT-1 | 9 | $0.999 \pm 0.0037$ |
| ANN2 | DT-2 | 11 | $0.996 \pm 0.0087$ |
| ANN3 | DT-3 | 8 | $0.938 \pm 0.0351$ |
| ANN4 | DT-4 | 10 | $0.995 \pm 0.0101$ |
| ANN5 | DT-5 | 9 | $0.979 \pm 0.0124$ |

**Table 4.7:** Overall NN classification performance over 10 validation sets, for each dataset with respective optimal features number for each one.

The classification results are very good, always above the level of the nineties, which immediately implies that all models are generalizing correctly, without going into an *over-fitting* situation. Corroborating the idea, Figure 4.9 represents the *learning curve* of *ANN4*.

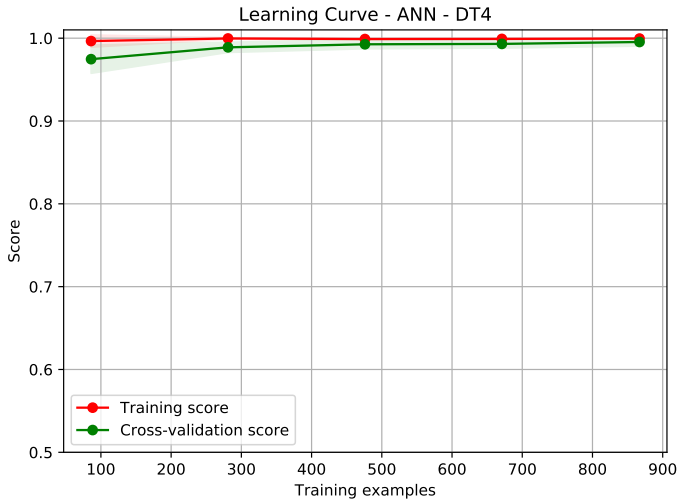



**Figure 4.9:** *Learning curve* for *ANN4*, showing how learning improves with samples number.

From the analysis of the graphical representation of the training score and *cross-validation* due to the increase in the number of samples, it is quickly confirmed that the model is generalizing correctly, even with few samples(e.g., 100), and that the increase in the number of observations network will not make the model generalize better, because the training score from the 700 samples overlap with the validation score. This simplicity of models that contemplate a single layer with *perceptrons* varying between 1 and 9, depending on the dataset, allows fast and correct convergence even with few data. With increasing complexity of models and inherent robustness, the high number of free parameters between the layers and the various *perceptrons* would require excessive amounts of data whereas the time for the model to generalize would increase substantially.

Another conclusion that can be drawn from the analysis of the various accuracies, presented on Table 4.7 is that the impact of *DT-0* segregation on the remaining datasets is insignificant. However, it is interesting to verify if the worst-accuracy model(i.e., *ANN3*) that generalizes from the *DT-3* dataset, when faced with new unseen data[1] presents better results than the model related to *DT-0* (i.e., *ANN0*), since it included data related to other attack vectors, in addition to the normal and specific data to the attack vector that is common. Thus, both methods were subjected to three equal *zero-day* tests, composed of 25 samples obtained from the exclusive observation of a GRE IP flood, following a *DYN-PW* dynamics, with a peak of 100 packets between periodic intervals of 30 seconds. This dynamic was chosen as an example based on the set that characterizes the data set related to this attack, which is a majority of attacks that follow a non-stealth dynamic(i.e., *DYN-CR*), which holds about 90% of the data that identifies the vector in question. The results are presented on the following Table.

[1]Also known as *zero-day* tests

| Zero-day | Model | PCA | Accuracy |
|---|---|---|---|
| Test 0 | ANN0 | 12 | 0.520 |
| Test 0 | ANN3 | 8 | 0.760 |
| Test 1 | ANN0 | 12 | 0.600 |
| Test 1 | ANN3 | 8 | 0.760 |
| Test 2 | ANN0 | 12 | 0.400 |
| Test 2 | ANN3 | 8 | 0.600 |

**Table 4.8:** *Zero-day* test performance for each model with respective ideal number of components. Since this is a one-time classification, it is not possible to provide a confidence interval.

This distance to the normal attack pattern with which the model had contact in the training phase allows to evaluate the model, as if it were in an environment in production where it is subject to a vast set of dynamics. From the analysis of the table it is possible to conclude that the *ANN3*, for this particular dynamic, performs slightly better than the model fed with all the observations of attacks, making this good in general, able to detect more anomalies, to the detriment of *ANN3* which is very good in the specific case of *AV-3*.

### 4.5.2 Decision Trees

In comparison to all data pipelines with NN present in the stage of ML, those that assume algorithms like decision tree and derivatives are invariant to monotonic transformations, as mentioned in Section 3.4.5.3 and therefore the feature scaling stage is omitted. All other stages prevail, thus constituting data pipelines with four phases.

Similar to the previous results generation process, this one also contemplated the hyper parameterization for the respective datasets of the various models, which were later evaluated not only on the *cross-validation* processes, also with 10 folds, but also over the test sets, being these unitary sets does not allow the calculation of the confidence interval. Table 4.9 places both metrics side by side for each of the models.

| Model | Dataset | PCA | Cross Val Accuracy | Test Accuracy |
|---|---|---|---|---|
| DTC6 | DT-0 | 12 | $0.915 \pm 0.0309$ | 0.923 |
| DTC7 | DT-1 | 9 | $0.972 \pm 0.0311$ | 0.974 |
| DTC8 | DT-2 | 11 | $0.968 \pm 0.0143$ | 0.959 |
| DTC9 | DT-3 | 8 | $0.914 \pm 0.0485$ | 0.882 |
| DTC10 | DT-4 | 10 | $0.953 \pm 0.0635$ | 0.981 |
| DTC11 | DT-5 | 9 | $0.906 \pm 0.0496$ | 0.875 |

**Table 4.9:** *Decision tree* classification performance for each dataset, with respective optimal number of features.

Despite the exhaustive search and tuning of the hyper parameters, compared to the NN there is a decrease of the accuracy common in all models that in the worst case scenario, for the dataset *DT-0* and *DT-5*, reaches a difference of 7%(e.g., *ANN5* to *DTC11*). However, a deeper interpretation of the model and its hyper parameters would be required to understand such differences in accuracies. Nevertheless, it does not compromise the veracity of the models,

which, like those presented in Section 4.5.1, are not *over-fitting*, since the depth of each one has been carefully selected, ranging from 6 to 15. Despite the lack of a standard for the depth of decision trees, once it varies according to the problem, it is possible to confirm by comparing the last two columns of the previous table. In general, 60 percent of the time, the accuracy of the test is greater than that obtained after *cross-validation*, allowing to confirm that the model generalizes correctly. Despite the 40 percent, the difference between the accuracies is minimal, indicating that the models are not too adjusted to the training data, not *over-fitting*, and have a very satisfactory behavior when faced with new test samples.

As discussed in Section 3.5.1, Figure 4.10 shows the normalized *confusion matrix* for both best and worst-case scenario, according to the test accuracy.

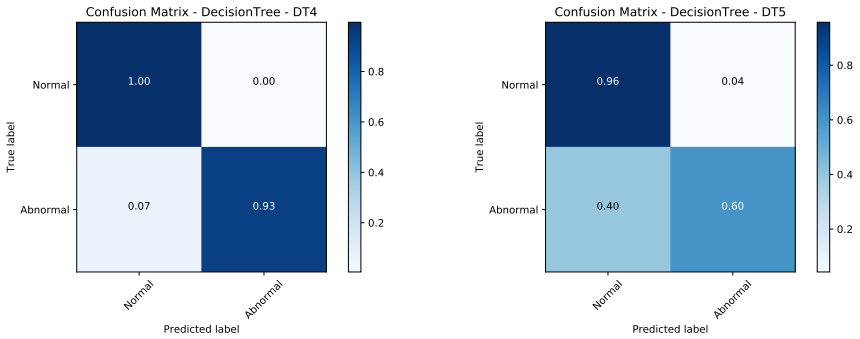



**Figure 4.10:** Normalized *confusion matrix* for *models 10* and *11* over the test set(i.e., 20 % of dataset *DT-4* and *DT-5* respectively).

Although both models differ not in the parameters but also in the datasets, and therefore are incomparable among them, it is interesting to understand the tendency to misclassification. On the left side, for the best case scenario, with regard to the samples that express the action of normal network activities, the model always correctly ranks, whereas in relation to the samples that identify the action of *AV-4* , in 7% of the times classified as normal, resulting in the presence of four *false negatives*(i.e., $0.07 * 58$, where 58 is the number of abnormal samples present on test set for *ANN4*). Regarding the worst case scenario, the trend remains in the classification of abnormal samples in normal, resulting in a higher ratio of *false negatives*, where the difference for *true negatives* is only 20%, among 36 correctly classified samples for 25 misinterpreted, given the total 62 abnormal samples present on the 20% of the *DT-5*. Compared to the *ANN5* that shares the same dataset, presenting a better performance, it is certain that the deficit is related to the algorithm used. Conversely, *DTC11* does not have a generalization degree as high as the previous model for the same amount of samples, as can be confirmed by the *learning curve*, shown in Figure 4.11.

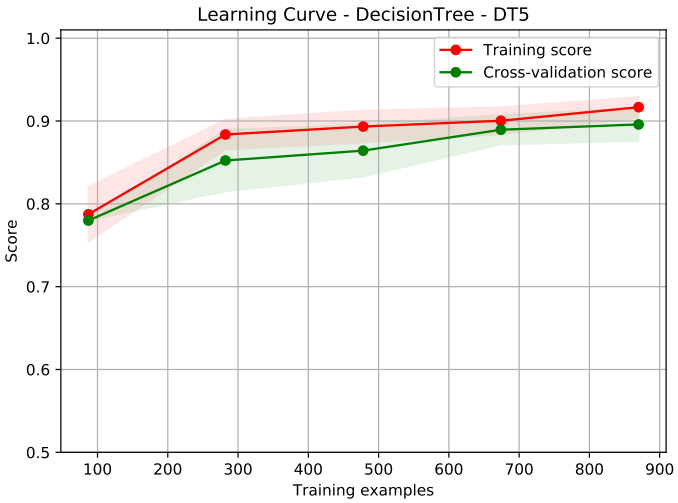



**Figure 4.11:** *Learning curve* for *DTC11*

Not surprisingly, although both lines are close, there is room for improvement with the introduction of more data corroborating the limitation of unbalanced dataset, raised in Section 4.4. Although this improvement has not been carried out, it is necessary to keep in mind the *generalization error* when introducing new data, which may bias the model incurring in *over-fitting*, as mentioned in Section 2.9.3.6.

### 4.5.3 Ensemble Methods

In both previous sections, in spite of the remarkable accuracies, it is possible to notice the performance differences between the classifiers fed by the *DT-3* and *DT-5* datasets and the other classifiers, ignoring those that based on the total set of observations. These differences can be explained by the simultaneous presence of the attack vectors *AV-3* and *AV-5*, respectively, with services that generate high amounts of traffic, such as file sharing(e.g., torrent), that allow the dissociation of the patterns inherent to the two attack vectors. While the NN as the decision trees are error-prone algorithms, the optimization is performed at the level of parameters and features, *Adaboost* and *Gradient-boost* in addition to this optimization, follow the boosting method allowing self-correction and thus recognition of these obfuscated patterns. Since both *Adaboost* and *Gradient-boost* are derived from decision trees, the requirements during the tuning of the hyper parameters related to *tree-specific* ones were transposed from the search process applied on the previous models. Moreover, both ensemble methods also inherit the feature scaling invariant and thus the similarity between data pipelines.

As discussed in Section 3.4.5.4, there are parameters that affect the boosting operation of the models(i.e., *boosting parameters*), which were also subject to a search process in order to maximize the classifiers performance. The research focused mainly on the learning rate and number of estimators, which effectively control the generalization and robustness of the models along with the computational resources underlying the training process.

Table 4.10 compiles the accuracies of both algorithms, after the *cross-validation* process, for all six datasets with respective number of ideal features for each one.

| Models | Dataset | PCA | Accuracy |
|---|---|---|---|
| ADA12 | DT-0 | 12 | $0.989 \pm 0.0066$ |
| ADA13 | DT-1 | 9 | $0.995 \pm 0.0115$ |
| ADA14 | DT-2 | 11 | $0.999 \pm 0.0042$ |
| ADA15 | DT-3 | 8 | $0.973 \pm 0.0187$ |
| ADA16 | DT-4 | 10 | $0.990 \pm 0.0115$ |
| ADA17 | DT-5 | 9 | $0.975 \pm 0.0170$ |

| Models | Accuracy |
|---|---|
| GDB18 | $0.985 \pm 0.0117$ |
| GDB19 | $0.997 \pm 0.0074$ |
| GDB20 | $1.000 \pm 0.0000$ |
| GDB21 | $0.977 \pm 0.0248$ |
| GDB22 | $0.993 \pm 0.0103$ |
| GDB23 | $0.978 \pm 0.0194$ |

**Table 4.10:** *Adaboost*, on the left, and *Gradient-boost*, on the right, classification performances, maintaining the dataset and PCA for both algorithms.

According to the performance of both classifiers, it is notable that models based on boosting operations outperform the models presented in the previous sections. Masked patterns are easily detected with algorithms based on weak learners, which despite being error-prone, perfectly adapt to the varied datasets, and underlying labeling strategies. For example, focusing on the *ADA13* analysis, the exceptional classification is reflected either in the learning curve or in the ROC curve, which antagonistically to the definition, take the form of a straight segment, as shown in Figure 4.12.

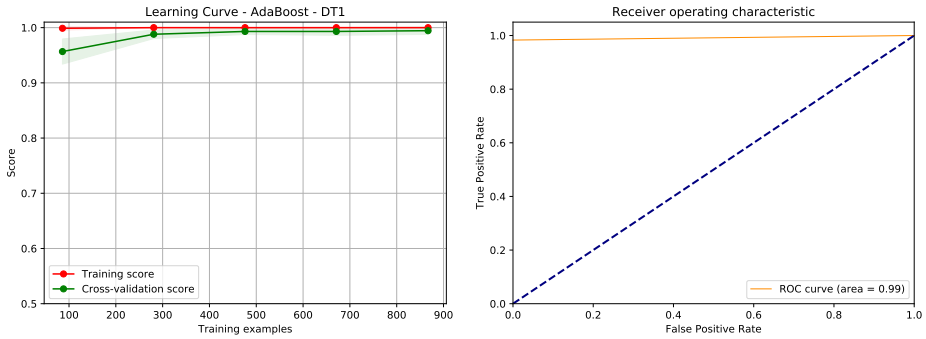



**Figure 4.12:** On the left *Adaboost* learning curve after *cross-validation* over the *DT-1* 80 % while on the left there is represented the ROC curve computed from the classification on the test set.

With respect to the learning curve, it assumes the ideal behavior avoiding altogether the high variance or high bias between the two scores, where the model converges with only 300 samples, and subsequent do not influence the prediction performance.

On the right side, the non-resemblance to a curve means that the classifier has completely distanced itself from the random reference represented by the dashed line, assuming a behavior very close to perfection, where the number of *false positives* and *negatives* tends to be zero, broadly classifying the new instances of the test set correctly.

A characteristic approach of these models to increase the robustness is the decrease of the learning rate and increase in the number of estimators proportionally, however, due to the

outstanding classification performances, it was decided to shift focus on the *zero-days* tests, which as mentioned in Section 3.5.2.

#### 4.5.3.1 *Zero-day* tests

As discussed in Section 3.5.2, although it is an optional process, its implementation is an added value to conclude about the overall solutions performance, forcing scenarios that are dissociated from those that are reproduced in the set of network observations that fed the models. Following an approach aimed at increasing the distance for highly intrusive dynamics, a number of network captures were collected for the first four attack vectors presented in Table 4.2, based on a *DYN-PW* dynamics, with several periods of silence, either periodic or following the exponential distribution, as mentioned in Section 4.1.2, without ruling out the simultaneity with other services. It is important to mention that all captures were carried out over a period of five minutes, thus resulting in 3000 sampling windows and therefore 25 observation windows, with a $\Delta$t of 1 minutes, as discussed in Section 4.3.

As discussed, the intrusiveness of an attack is largely defined by the combination of two variables: packet throughput and the duration of silence periods. Looking first at the dynamics with the highest packet throughput per activity period, the first(i.e., *DYN-PW-1*) was implemented by setting the debt at 300, varying the duration of the silence periods following Equation 4.1 with a scalar parameter of 0.1. The inherent behavior was reported concurrently with YouTube(i.e., the only service considered for sake of simplicity), which as in all subsequent dynamics have been commonly implemented and recorded for the respectively four attack vectors.

Intending to elucidate the duration of silence periods, Figure 4.13 shows an example of its distribution in relation to the number of periods. It should be noted that in all the dynamics implemented, the maximum ceiling for the duration of the silence periods was 60 seconds, assuring the presence, even if minimal, of packets referring to attack vectors in the observations windows.

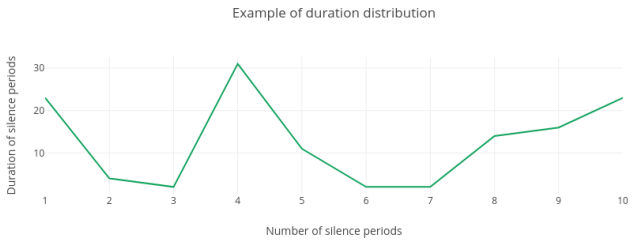



**Figure 4.13:** Distribution example of the duration of each period of silence resulting from the exponential distribution with scalar equal to 0.1.

Compared to the highly intrusive *DYN-CR* dynamic characterized by the throughput of about 180 packets per second, the significant reduction in the total number of packets(i.e., typically 11000 packets per minute) and the insertion of periods of silence with reasonable durations, allows for the decrease in the intrusivity spectrum also due to the obfuscation with

intrinsic Youtube traffic. Applied dynamics to the *AV-1* and *AV-4*, Table 4.11 shows the performance differences between Adaboost and Gradient-boost models.

| Model | Attack vector | Dynamic | PCA | Accuracy |
|---|---|---|---|---|
| ADA13 | AV-1 | DYN-PW-1 | 9 | $1.00 \pm 0.0000$ |
| GDB19 | AV-1 | DYN-PW-1 | 9 | $0.512 \pm 0.1610$ |
| ADA16 | AV-4 | DYN-PW-1 | 10 | $0.360 \pm 0.1700$ |
| GDB22 | AV-4 | DYN-PW-1 | 10 | $0.720 \pm 0.1670$ |

**Table 4.11:** Adaboost and Gradient-boost classification performance for two attack vector, both with same dynamic. Extensive results available in Table A.1 and A.2.

Due to the inherent abstraction of the use of an exponential distribution it is not possible to compare the results impartially between attack vectors, since in the *AV-1* it can assume periods of silence that are always high, increasing the stealth of the attack even more, in detriment of the values obtained for the other. However, as for the classifiers, each pair is confronted with the same data, being on an equal footing. According to the results it is clear that there is no consensus, since for each attack vector with the same dynamics, there is a more flexible model than the other, producing better results. However it is possible to state that the dissociation of attack traffic on Youtube is still detectable. Increasing the distance from the intrusive end, by reducing the packet throughput by a third of the previous dynamics(i.e., 300 - 300/3) and maintaining the scalar parameter of the exponential distribution, it is observed that for the *AV-1* attack vector, passing the same data to the two models, 13 and 19, the results of *GDB19*($0.912 \pm 0.0520$) are extremely better than that of model *13*($0.080 \pm 0.0429$), concluding that for this attack vector, the model based on the Adaboost algorithm is more extensive and produces better results in unseen dynamics. Extended results are present in Table A.3.

In order to compare the effect of the patterns dissociation in legitimate traffic, another dynamics was implemented(i.e., *DYN-PW-2*) resulting from the variation of the scale parameter present in the equation of the exponential distribution in question. The parameter was set at 0.02, with a difference of 0.08 for the previous dynamics. From the various periods of silence obtained, it is possible to conclude that the lower the scalar, the longer the periods will be, and therefore the greater the stealth of the dynamic. Table 4.12 shows the results of the influence of the presence of Youtube activity in the two attack vectors detectability, *AV-1* and *AV-3* namely.

| Model | Attack vector | Dynamic | PCA | Youtube | Accuracy |
|---|---|---|---|---|---|
| ADA13 | AV-1 | DYN-PW-2 | 9 | Yes | $0.136 \pm 0.1390$ |
| ADA13 | AV-1 | DYN-PW-2 | 9 | No | $0.912 \pm 0.1370$ |
| ADA15 | AV-3 | DYN-PW-2 | 8 | Yes | $0.256 \pm 0.0807$ |
| ADA15 | AV-3 | DYN-PW-2 | 8 | No | $0.952 \pm 0.0938$ |

**Table 4.12:** Adaboost classification performance including dissociation of patterns in Youtube traffic, for two attack vectors. Extended results are present in Table A.4 and A.5.

Not surprisingly, in spite of the exponential distribution introducing abstraction and

randomness, by correlation of the results of the two models, for the *zero-day* tests generated with the dynamics *DYN-PW-2* with simultaneity with Youtube, in half of the samples, it is possible to confirm the fingerprints dissimilation of each attack vector in legitimate traffic.

Moving towards stealth, the ceiling of the maximum number of packets sent in each period of activity was again reduced, to 100 keeping the same parameter scale at 0.02, constituting the dynamic *DYN-PW-3*, characterized by the lowest scale parameter of all implemented. In this way, allowing to compare the variation of the duration of the silence periods, Figure 4.14 represents the contrast of the silences duration in detriment of the scale parameter of the exponential distribution obtained from Equation 4.1.

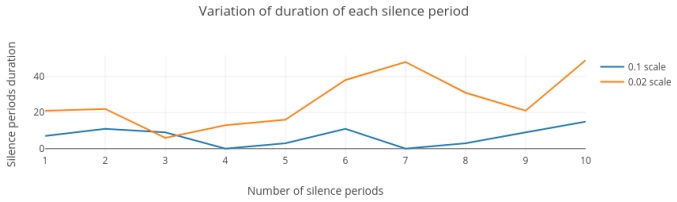



**Figure 4.14:** Example of the scale parameter variation over the resultant silence periods duration, being high stealth when assumes lower values.

This dynamic given its sneaky character, was not co-captured with the presence of other services. Thus, the results presented in the following table correspond to the 125 samples captured for each attack vector with *DYN-PW-3* dynamics.

| Model | Attack vector | Accuracy |
|---|---|---|
| GDB19 | AV-1 | $0.128 \pm 0.1200$ |
| GDB20 | AV-2 | $0.080 \pm 0.1570$ |
| GDB21 | AV-3 | $0.544 \pm 0.1690$ |
| GDB22 | AV-4 | $0.024 \pm 0.0471$ |

| Models | Accuracy |
|---|---|
| ADA13 | $0.824 \pm 0.1230$ |
| ADA14 | $0.000 \pm 0.0000$ |
| ADA15 | $1.000 \pm 0.0000$ |
| ADA16 | $0.000 \pm 0.0000$ |

**Table 4.13:** *Gradient-boost* and *Adaboost* performances comparison, from left to the right, maintaining the *zero-day* samples per row. Extended results are provided in Section A.1.

As mentioned, given the ambiguity resulting from the various network captures with periods of inconsistent silences, it is possible to confirm the decreasing tendency of the accuracies. As between the vectors of attack an unbiased comparison, as indicted, it is verified that for the *AV-1* the model based on the *Adaboost* is more flexible and maintains the bar of high accuracies even for stealth dynamics. However, for the vectors of attack *AV-2* and *AV-4* the results are not satisfactory, reflecting the inflexibility for scenarios more distant from the intrusive ones contemplated in the training. A possible solution to combat this high variability of results is the training with more stealthy dynamics, allowing the acquisition of network anomaly patterns from a range of larger and more diverse observations, thus contributing to generalization in less intrusive dynamics.

Up to this point, only models characterized by a learning process based on data belonging to a single attack vector have been confronted. Changing the focus to the models that were fed

with the 2545 network observations(i.e., *model* 12 and *18* respectively), Table 4.14 compiles several results for the dynamic previously described.

| Model | Attack vector | Dynamic | Accuracy |
|---|---|---|---|
| GDB18 | AV-1 | DYN-PW-3 | $0.950 \pm 0.0613$ |
| GDB18 | AV-2 | DYN-PW-3 | $0.920 \pm 0.0990$ |
| GDB18 | AV-3 | DYN-PW-3 | $0.952 \pm 0.0938$ |
| GDB18 | AV-4 | DYN-PW-3 | $0.360 \pm 0.0990$ |

| Models | Accuracy |
|---|---|
| ADA12 | $0.688 \pm 0.1930$ |
| ADA12 | $0.616 \pm 0.1520$ |
| ADA12 | $0.776 \pm 0.1800$ |
| ADA12 | $0.544 \pm 0.0877$ |

**Table 4.14:** Comparison between the results obtained using the same *zero-day* tests over two different ML algorithms, on the left *Gradient-boost* and on the right *Adaboost*. Extended results available in Section A.1.1.

Challenging models with stealthy dynamics, similar to human behavior, are visible good results, even for the data that the models discussed previously presented near zero accuracies. Among models, it is also possible to conclude that *Gradient-boost* is the most robust model, producing better results in three quarters of the time, even providing better results than GDB19, until then the best classifier, with respect to the *AV-1* attack vector. This contrast of results between models fed with different datasets shows that models that based on larger datasets, aggregating all the vectors of attacks in the label of anomalies, present a higher degree of generalization.

Another approach to revalidate high flexibility and excellent generalization is the addition of low throughput dynamics. These were implemented by setting the packet flow rate at 20 and 50 per activity period, intervals with periods of silence lasting 10 and 60 seconds. These low-debt dynamics, contrast with the previous ones, not only by the decrease in the intrusivity spectrum, but also by the introduction of periodicity. Since the low packet rate with duration of the high silence periods for an observation window with a $\Delta$t value of one minute, already makes the dynamics highly stealthy, the increase of the randomness inherent to the use of the exponential distribution as well as the simultaneity with other services would introduce too much dispersion, making impossible the conscious analysis of the several results, ordered by stealth in Table 4.15.

| Model | Attack vector | Pulse wave peak | Silences duration | Accuracy |
|---|---|---|---|---|
| ADA12 | AV-2 | 50 packets | 10 seconds | 1.000 |
| ADA12 | AV-2 | 20 packets | 10 seconds | 1.000 |
| ADA12 | AV-2 | 50 packets | 60 seconds | 0.760 |
| ADA12 | AV-2 | 20 packets | 60 seconds | 0.680 |

**Table 4.15:** *Adaboost* model ADA12 results for low-rate ACK flood.

Not surprisingly, the results follow what was expected, since in an observation window with a $\Delta$t of one minute, the greater the packets number the greater the accuracy. In contrast, when the same test samples are applied to *GDB18*, which is based on the *Gradient-boost* classifier, the results become inconsistent. Strangely it only detects when the silence period assumes the duration of 60, when it was expected to present worse accuracies in this case due to the smaller presence of packets in the observation window. In order to avoid any errors

inherent in the network captures in question, the same dynamics were applied to the AV-4. As can be seen from Table 4.14, this attack vector presents, in general, better results when the classifier is *Adaboost*. As shown in Table 4.16, the inconsistencies are also recurrent for other test samples, making *GDB18* less reliable for low-throughput packet dynamics.

| Model | Attack vector | Pulse wave peak | Silences duration | Accuracy |
|---|---|---|---|---|
| GDB18 | AV-4 | 50 packets | 10 seconds | 0.000 |
| GDB18 | AV-4 | 20 packets | 10 seconds | 0.000 |
| GDB18 | AV-4 | 50 packets | 60 seconds | 0.760 |
| GDB18 | AV-4 | 20 packets | 60 seconds | 0.760 |

**Table 4.16:** *Gradient-boost* model GDB18 results for low-rate DNS-query flood.

Based on the dynamics more distant from the intrusive end of the spectrum, corroborating with the results for the dynamics previously discussed, it is concluded that the classifiers that contemplated all network observations, despite the inherent complexity of the various patterns and their aggregation tend to be more robust and flexible, even in dynamics were from the comfort zone, corroborating the excellent generalization to detect the various anomalies. However, the model based on *Gradient-boost* method, presents in general, a high overall accuracy for all kinds of inherent anomalies or dynamics.

To conclude, by following the presented methodologies in the previous chapter, this proof of concept is a corroboration of them, with good results and space to improve. The resulting models demonstrated not only a correct selection of features but also a high degree of generalization along with flexibility when confronted with behaviors that somehow diverge from those present in the training phase. Despite this, it is important to note that all data were obtained in a controlled environment, and in order to make the models more robust and comprehensive.

CHAPTER 5

# Conclusion and Future Work

As discussed in Chapter 2, being resource availability and network security two recurrent topics, both are constantly being focused in the spotlight of Internet giants resulting in major research investments. New botnets and attack vectors that exploit new vulnerabilities, misuse new protocols and avoid actual detection techniques are constantly emerging, being network activity the central point to such detection. Inferring behavioral patterns is to understand the legitimacy of an action or use of a service in accordance with what is expected, constituting only the first layer of awareness. Identifying the network threat and traceback to determine the source of the threat in the network topology ensures the application of a whole new level of countermeasures to mitigate an attack, such as automatic responses including traffic segregation, or device isolation.

With regard to the network topology adopted, described in Section 4.1, due to the limitation of resources during the dissertation, it was not possible to strictly replicate the topology of an enterprise network. It would be highly advantageous to have a set of VLANs, with distinct employee groups associated with different behaviors, a set of typical services from a corporate scenario, with access to datacenters, cloud and internal services, without ruling out the existence of all devices common to any larger network such as smart TV providing IPTV content, surveillance cameras and other IoT devices. Inherently to these services and users there are several standards that could not be accounted for, even if it was highly desirable. As for the anomalies contemplated, their real character with past evidence emphasizes the importance of them and the preponderant role they play in the development of any defense solution. Although only five attack vectors with several associated dynamics have been selected, it is possible and highly advisable to extend this set, covering a greater number of anomalies with a view not only to the completeness of the solution but also to the detection of zero-day attacks exploiting unknown faults, but which may resemble known patterns contributing to detection in improbable scenarios. The number of services to operate simultaneously with the attack vectors was limited, and in the future must be analyzed and inherent pattern to each services and according to the similarity to the various vectors of

attack, tend to select those that best obfuscate anomalous traffic, thus increasing the detection difficulty in order to enhance to the overall detection performance of the solution. This criterion unfortunately was not considered, which may justify, even with a low percentage, the detectability of the attack vectors simultaneously with Youtube, as shown in Table 4.12. Of course, if it were simultaneously with file sharing traffic (i.e., torrent), the results would be worse because of the similarity between the attack vector (i.e., AV-1) and the subsequent traffic of service utilization, which tends to generate excessive amounts of traffic, contributing to the patterns dissociation.

On data collection level described in 4.1, in spite of the topology simplicity and the lack of requirements for the network sniffer placement, the concept of the module was thought for scalability, assuming a generic nature that allows adaptation in future use cases, for higher layers. However, there is a performance bottleneck, requiring storage resources directly proportional to the amount of traffic on the network. Collecting all packet information and storing it fits solely for study purposes, in order to contemplate methods for inferring network patterns. Although packet processing has been accelerated, supported by current benchmarks, it is important to couple the collection and processing processes, not only to avoid the use of excessive storage, by adopting filtering techniques when collecting the data and storing it in a simple metadata structure as suggested in Section 3.2.1, but also by saving time by contributing to the real-time nature of the solution. Knowing that all processing requires resources, another possible approach involves sending collected, pre-processed or raw network data to a cloud for further processing and subsequent operations.

Regarding the features engineering discussed in Section 4.3, from the network, only statistic data were designed, which proved sufficient to detect anomalies. Without distancing the network domain, a direct approach to improving the patterns representation is the inclusion of metrics associated with the use of protocols and the aggregation of metrics per data streams:

- Analyze the TCP flags ratio within a network stream (i.e., TCP session);
- Accounting for failed sessions, when there is no response after a session establishment attempt;
- TCP sessions duration and byte transfer;
- Frequency of GRE headers, ratios with its volumes;
- Correlate volumes with services and IPs.

Other suggestions include the addition of metrics that allow you to define periodic and non-periodic events (e.g., using wavelet analyzes), as well as attributes to detect traffic burst and its duration, applying over them the considered statistical processing including counting, means, medians, and variances. It would also be appropriate to measure the dispersion of the durations of the periods of silence, since it provides information more valuable than the simple mean as contemplated in the solution. Any inclusion of features must contemplate the balancing between the overhead computational resources for the generation of features and the real-time decision character of the solution. Regarding the duration of the observation windows, it would be interesting to decrease it sequentially and compute the various results, exhaustively, with a view to decreasing the response time without compromising the anomalies detectability.

The two labeling strategies, and underlying datasets, detailed in Figure 4.3, bifurcate the results showing that the same data can be viewed differently, with associated advantages and disadvantages, without departing from the scope of the solution.

From the dataset overview to the deep analysis of features that compose the various network observations, present in Section 4.4, the explicit differences between the footprints of each attack vector considered on this dissertation, proves the existence of enough discriminators to detect such anomalies. This perception is corroborated by the good generalization and results provided by all ML algorithms, presented in Section 4.5. However, it is necessary to emphasize the space to improve the parameterization of the NN, which climb exceptionally well being an added value for adoption in a future scenario with larger amounts of data. Without obviously discarding the ensemble methods, which due to their boosting operation mode, presented the best performances for all datasets, available on Table 4.10). Regarding the zero-days, its use allowed to evaluate the flexibility of each model in extremely stealth scenario and formalize specific use cases on the applicability of each model. Widely the model that best balances the commitment of detecting anomalies and producing false positives, on both intrusive and stealth scenarios, is the model based on the Gradient-boost algorithm, which was fed with the totality of network observations. However, it would be highly rewarding for the performance and reliability of the models in scenarios with attack dynamics similar to human behaviors, adding the various samples obtained in the zero day Section 4.5.3.1, to the respective datasets, increasing the data pool and underlying distribution of dynamics.

Another approach to the final solution follows the order from general to specific, where the models would initially be arranged according to a pipeline structure. The first layer would correspond to the general detection of the anomaly, using the best model (i.e., is not able to distinguish the type of anomaly). Later, the network anomaly would be identified in the second layer by a set of models, each one in charge of the detection of a type of anomaly, like the models that contemplated solely an attack vector, through a voting process, as it exists in the OvA strategy, detailed in Section 3.4.4. Finally, the last layer would be responsible for activating not only traceback mechanisms but also counteracting the compromised machine and active attack.

On the whole, Chapter 4 confirmed that it is possible and highly reliable to correlate anomalies with the network activity, and therefore contribute to the increase of network awareness and security. Broadly the raw data is collected, refined into meaningful information that is ultimately converted into knowledge, and in light of this, the dataset needs an exhaustive evaluation with real-world testing, including for example flash-crowd events.

DDoS attacks require a stable and trustworthy defense solution. This dissertation has presented a crucial building block of this solution, being the developed methodologies a OS system architecture that provides a fast and reliable response. It can offer an excellent performance, even for autonomous counteract operations, enhancing it when integrated and deployed with other solutions improving the security and inherent resources availability of a network.

APPENDIX A

# Zero-day Extended Results

| Sample Number | Dynamic | PCA | *GDB19* | *ADA13* |
|---|---|---|---|---|
| 1 | DYN-PW-1 | 9 | 0.520 | 1.000 |
| 2 | DYN-PW-1 | 9 | 0.680 | 1.000 |
| 3 | DYN-PW-1 | 9 | 0.600 | 1.000 |
| 4 | DYN-PW-1 | 9 | 0.560 | 1.000 |
| 5 | DYN-PW-1 | 9 | 0.200 | 1.000 |

**Table A.1:** *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-1*, which have a peak of 300 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1.

| Sample Number | Dynamic | PCA | *GDB22* | *ADA16* |
|---|---|---|---|---|
| 1 | DYN-PW-1 | 10 | 0.920 | 0.520 |
| 2 | DYN-PW-1 | 10 | 0.840 | 0.440 |
| 3 | DYN-PW-1 | 10 | 0.560 | 0.320 |
| 4 | DYN-PW-1 | 10 | 0.800 | 0.480 |
| 5 | DYN-PW-1 | 10 | 0.480 | 0.040 |

**Table A.2:** *Gradient-boost*(i.e., *GDB22*) and *Adaboost*(i.e., *ADA16*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-1*, which have a peak of 300 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1.

| Sample Number | Attack vector | PCA | *GDB19* | *ADA13* |
|---|---|---|---|---|
| 1 | AV-1 | 9 | 0.920 | 0.120 |
| 2 | AV-1 | 9 | 1.000 | 0.120 |
| 3 | AV-1 | 9 | 0.840 | 0.080 |
| 4 | AV-1 | 9 | 0.880 | 0.080 |
| 5 | AV-1 | 9 | 0.920 | 0.080 |

**Table A.3:** *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following a dynamic with a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.1.

| Sample Number | Dynamic | PCA | *ADA13* - Youtube | *ADA13* |
|---|---|---|---|---|
| 1 | DYN-PW-2 | 9 | 0.360 | 1.000 |
| 2 | DYN-PW-2 | 9 | 0.080 | 0.640 |
| 3 | DYN-PW-2 | 9 | 0.000 | 1.000 |
| 4 | DYN-PW-2 | 9 | 0.000 | 1.000 |
| 5 | DYN-PW-2 | 9 | 0.240 | 0.920 |

**Table A.4:** *Adaboost*(i.e., *ADA13*) classification performances over samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-1*, which have a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. The action of the attack vector was initially captured simultaneously with Youtube activity, and later in isolation.

| Sample Number | Dynamic | PCA | *GDB19* - Youtube | *GDB19* |
|---|---|---|---|---|
| 1 | DYN-PW-2 | 8 | 0.200 | 1.000 |
| 2 | DYN-PW-2 | 8 | 0.400 | 1.000 |
| 3 | DYN-PW-2 | 8 | 0.240 | 1.000 |
| 4 | DYN-PW-2 | 8 | 0.160 | 1.000 |
| 5 | DYN-PW-2 | 8 | 0.280 | 0.760 |

**Table A.5:** *Gradient-boost*(i.e., *GDB19*) classification performances over samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-1*, which have a peak of 200 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02. The action of the attack vector was initially captured simultaneously with Youtube activity, and later in isolation.

## A.1 DYN-PW-3

| Sample Number | Dynamic | PCA | *GDB19* | *ADA13* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 9 | 0.000 | 0.760 |
| 2 | DYN-PW-3 | 9 | 0.080 | 0.800 |
| 3 | DYN-PW-3 | 9 | 0.360 | 1.000 |
| 4 | DYN-PW-3 | 9 | 0.120 | 0.640 |
| 5 | DYN-PW-3 | 9 | 0.080 | 0.920 |

**Table A.6:** *Gradient-boost*(i.e., *GDB19*) and *Adaboost*(i.e., *ADA13*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB20* | *ADA14* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 11 | 0.000 | 0.000 |
| 2 | DYN-PW-3 | 11 | 0.000 | 0.800 |
| 3 | DYN-PW-3 | 11 | 0.000 | 0.000 |
| 4 | DYN-PW-3 | 11 | 0.000 | 0.000 |
| 5 | DYN-PW-3 | 11 | 0.480 | 0.000 |

**Table A.7:** *Gradient-boost*(i.e., *GDB20*) and *Adaboost*(i.e., *ADA14*) classification performances over the same samples obtained from the *AV-2*(i.e., TCP-ACK flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB21* | *ADA15* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 8 | 0.400 | 1.000 |
| 2 | DYN-PW-3 | 8 | 0.720 | 1.800 |
| 3 | DYN-PW-3 | 8 | 0.600 | 1.000 |
| 4 | DYN-PW-3 | 8 | 0.680 | 1.000 |
| 5 | DYN-PW-3 | 8 | 0.240 | 1.000 |

**Table A.8:** *Gradient-boost*(i.e., *GDB21*) and *Adaboost*(i.e., *ADA15*) classification performances over the same samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB22* | *ADA16* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 10 | 0.000 | 0.000 |
| 2 | DYN-PW-3 | 10 | 0.000 | 0.800 |
| 3 | DYN-PW-3 | 10 | 0.000 | 0.000 |
| 4 | DYN-PW-3 | 10 | 0.000 | 0.000 |
| 5 | DYN-PW-3 | 10 | 0.000 | 0.120 |

**Table A.9:** *Gradient-boost*(i.e., *GDB22*) and *Adaboost*(i.e., *ADA16*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

### A.1.1 DYN-PW-3 applied to ADA12 and GDB18

| Sample Number | Dynamic | PCA | *GDB18* | *ADA12* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 9 | 1.000 | 0.720 |
| 2 | DYN-PW-3 | 9 | 1.080 | 0.360 |
| 3 | DYN-PW-3 | 9 | 1.000 | 0.840 |
| 4 | DYN-PW-3 | 9 | 0.920 | 0.600 |
| 5 | DYN-PW-3 | 9 | 0.840 | 0.920 |

**Table A.10:** *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-1*(i.e., TCP-SYN flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB18* | *ADA12* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 11 | 1.000 | 0.640 |
| 2 | DYN-PW-3 | 11 | 1.080 | 0.640 |
| 3 | DYN-PW-3 | 11 | 1.000 | 0.760 |
| 4 | DYN-PW-3 | 11 | 0.840 | 0.720 |
| 5 | DYN-PW-3 | 11 | 0.760 | 0.320 |

**Table A.11:** *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-2*(i.e., TCP-ACK flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB18* | *ADA12* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 8 | 1.000 | 0.600 |
| 2 | DYN-PW-3 | 8 | 1.000 | 1.000 |
| 3 | DYN-PW-3 | 8 | 1.000 | 0.880 |
| 4 | DYN-PW-3 | 8 | 1.000 | 0.880 |
| 5 | DYN-PW-3 | 8 | 0.760 | 0.520 |

**Table A.12:** *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-3*(i.e., GRE flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

| Sample Number | Dynamic | PCA | *GDB18* | *ADA12* |
|---|---|---|---|---|
| 1 | DYN-PW-3 | 10 | 0.400 | 0.960 |
| 2 | DYN-PW-3 | 10 | 0.200 | 0.880 |
| 3 | DYN-PW-3 | 10 | 0.480 | 1.000 |
| 4 | DYN-PW-3 | 10 | 0.280 | 0.840 |
| 5 | DYN-PW-3 | 10 | 0.600 | 0.920 |

**Table A.13:** *Gradient-boost*(i.e., *GDB18*) and *Adaboost*(i.e., *ADA12*) classification performances over the same samples obtained from the *AV-4*(i.e., DNS-query flood) action following *DYN-PW-3*, which have a peak of 100 packets intervals with silences according to Equation 4.1, with a scalar parameter of 0.02.

# References


[1] RSA, *Security and Privacy of Machine Learning | USA 2018 | RSA Conference*, 2018. [Online]. Available: https://www.rsaconference.com/events/us18/agenda/sessions/11533-security-and-privacy-of-machine-learning (visited on 09/13/2018).

[2] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems (part I.A)", no. October, pp. 2–4, 2000.

[3] Infosec, *CIA Triad*, 2017. [Online]. Available: http://resources.infosecinstitute.com/cia-triad/%7B%5C#%7Dgref%20http://resources.infosecinstitute.com/category/certifications-training/cissp/domains/security-and-risk-management/the-security-cia-triad/%7B%5C#%7Dgref (visited on 02/14/2018).

[4] L. Gallon and P. Owezarski, "Network Security and DoS Attacks 0.", Tech. Rep. April, 2005, pp. 1–24.

[5] T. Penttinen, "Distributed Denial-of-Service Attacks in the Internet", PhD thesis, 2005, p. 148. [Online]. Available: https://jyx.jyu.fi/dspace/bitstream/handle/123456789/12370/URN%7B%5C_%7DNBN%7B%5C_%7Dfi%7B%5C_%7Djyu-200662.pdf?seque...

[6] V. A. Eds and D. Ferrari, *Context-Aware Systems and Applications*. 2015, vol. 128, ISBN: 978-3-319-05938-9. DOI: 10.1007/978-3-319-05939-6. [Online]. Available: http://link.springer.com/10.1007/978-3-319-05939-6.

[7] Karen Hao, *A cryptocurrency-mining virus is spreading through Facebook messenger — Quartz*, 2017. [Online]. Available: https://qz.com/1168160/a-cryptocurrency-mining-virus-is-spreading-through-facebook-messenger/ (visited on 02/13/2018).

[8] P. Paganini, *WannaMine, the sophisticated crypto miner that spreads via NSA EternalBlue exploit-Security Affairs*, 2018. [Online]. Available: http://securityaffairs.co/wordpress/68518/malware/wannamine-nsa-eternalblue.html (visited on 02/13/2018).

[9] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms", *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 39, 2004, ISSN: 01464833. DOI: 10.1145/997150.997156. arXiv: arXiv:1011.1669v3. [Online]. Available: http://portal.acm.org/citation.cfm?doid=997150.997156.

[10] J. Mirkovic, "D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks", PhD thesis, University of California, 2003, p. 396.

[11] K. Eaton, *How One Second Could Cost Amazon 1.6 Billion In Sales*, 2012. [Online]. Available: https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales%20http://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales (visited on 02/14/2018).

[12] T. Bauer, *The world got impatient; one second of slower load time can cost Amazon $1.6 billion across a year | The Context Of Things*, 2014. [Online]. Available: http://thecontextofthings.com/2014/06/28/amazon-and-impatience/ (visited on 02/14/2018).

[13] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems", *ACM Computing Surveys*, vol. 39, no. 1, 3–es, 2007, ISSN: 03600300. DOI: 10.1145/1216370.1216373. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1216370.1216373.

[14] U. Akyazi, "Distributed Intrusion Detection using Mobile Agents against DDoS Attacks", *2008 23rd International Symposium on Computer and Information Sciences*, 2008.

[15] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions", *The Computer Journal, Oxfordjournals*, vol. 57, no. 4, pp. 537–556, 2014, ISSN: 0010-4620. DOI: 10.1093/comjnl/bxt031. [Online]. Available: http://comjnl.oxfordjournals.org/content/early/2013/03/28/comjnl.bxt031%7B%5C%%7D5Cnhttp://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/bxt031.

[16] T. Gunasekhar, K. Rao, P. Saikiran, and P. Lakshmi, "A Survey on Solutions to Distributed Denial of Service Attacks", *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5, no. 2, pp. 2373–2376, 2014, ISSN: 03722112. DOI: 10.1.1.77.1031. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-69249150102%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=3aa1db7515b67a5b4b1cb6be59df8ea4.

[17] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns", *2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, pp. 952–957, 2011, ISSN: 10636692. DOI: 10.1109/INFCOMW.2011.5928950.

[18] C. Science and S. Engineering, "Techniques to Differentiate DDOS Attacks from Flash Crowd", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, pp. 295–299, 2013. [Online]. Available: http://www.ijarcsse.com/docs/papers/Volume%7B%5C_%7D3/6%7B%5C_%7DJune2013/V3I5-0442.pdf.

[19] S. Lin and T.-c. Chiueh, "A Survey on Solutions to Distributed Denial of Service Attacks", *RPE report*, vol. 37, no. 7, pp. 1562–1570, 2006, ISSN: 03722112. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-69249150102%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=3aa1db7515b67a5b4b1cb6be59df8ea4.

[20] K. M. Prasad, A. R. M. Reddy, and K. V. Rao, "DoS and DDoS Attacks: Defense, Detection and TracebackMechanisms -A Survey", *Global Journal of Computer Science and Technology: E Network, Web & Security*, vol. 14, no. 7, p. 19, 2014.

[21] Arbor Networks, "Worldwide Infrastructure Security Report", Tech. Rep., 2016.

[22] D. Anstee, C. Chui, P. Bowen, and G. Sockrider, "Worldwide Infrastructure Security Report", vol. XII, no. 2017 Volume XII, 2017. [Online]. Available: https://www.arbornetworks.com/arbor-networks-12th-annual-worldwide-infrastructure-security-report-finds-attacker-innovation-and-iot-exploitation-fuel-ddos-attack-landscape.

[23] NETSCOUT SYSTEMS, "Insight Into The Global Threat Landscape", Tech. Rep., 2018, p. 93.

[24] Arbor Networks, "Worldwide Infrastructure Security Report", Tech. Rep., 2014.

[25] F. Y. Rashid, *Sony Data Breach Was Camouflaged by Anonymous DDoS Attack*, 2011. [Online]. Available: http://www.eweek.com/security/sony-data-breach-was-camouflaged-by-anonymous-ddos-attack (visited on 02/23/2018).

[26] INCAPSULA, *DDoS for Hire | Booter, Stresser and DDoSer | Incapsula*. [Online]. Available: https://www.incapsula.com/ddos/booters-stressers-ddosers.html (visited on 07/10/2018).

[27] A. Mohiuddin, M. A. Uddin, and P. G. M. Someswar, "Design and Development of an effective DDoS Shield to implement a well secured Defense System against vulnerable attacks", *International Journal of Advancements in Research & Technology*, vol. 3, no. 3, pp. 31–52, 2014.

[28] M. D. Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of DDoS-Capable IoT Malwares", *Computer Science and Information Systems*, vol. 11, pp. 807–816, 2017. DOI: 10.15439/2017F288.

[29] K. J. Houle, G. M. Weaver, N. Long, and R. Thomas, "Trends in Denial of Service Attack Technology", *October*, vol. 2008, no. October, pp. 0–20, 2001. [Online]. Available: https://resources.sei.cmu.edu/asset%7B%5C_%7Dfiles/WhitePaper/2001%7B%5C_%7D019%7B%5C_%7D001%7B%5C_%7D52491.pdf%20http://www.cert.org/archive/pdf/DoS%7B%5C_%7Dtrends.pdf.

[30] A. Bender, "An Accountability Architecture for the Internet Adam Bender , Doctor of Philosophy , 2010 Bobby Bhattacharjee Department of Computer Science", PhD thesis, 2010, p. 156. [Online]. Available: https://drum.lib.umd.edu/handle/1903/11194.

[31] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet denial of service : attack and defense mechanisms*. Prentice Hall, 2005, p. 372, ISBN: 0131475738.

[32] D. M. Kienzle and M. C. Elder, "Recent worms: a survey and trends", *Proceedings of the 2003 ACM workshop on Rapid malcode*, pp. 1–10, 2003. DOI: 10.1145/948187.948189. [Online]. Available: http://vx.org.ua/lib/pdf/Recent%20Worms:%20A%20Survey%20and%20Trends.pdf.

[33] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control", *Proceedings - Conference on Local Computer Networks, LCN*, no. December, pp. 195–202, 2006, ISSN: 0742-1303. DOI: 10.1109/LCN.2006.322100.

[34] C. Cimpanu, *Satori Botnet Is Now Attacking Ethereum Mining Rigs*, 2018. [Online]. Available: https://www.bleepingcomputer.com/news/security/satori-botnet-is-now-attacking-ethereum-mining-rigs/ (visited on 02/21/2018).

[35] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS Attacks: Trends and Challenges", *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2242–2270, 2015, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2457491.

[36] D. K. Bhattacharyya and J. K. Kalita, *DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance*. CRC Press, 2016, p. 311, ISBN: 9781498729659.

[37] R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Computer Networks", *Computer Networks*, vol. 57, pp. 378–403, 2013. DOI: 10.1016/j.comnet.2012.07.021.

[38] F.-h. Hsu, C.-w. Ou, Y.-l. Hwang, Y.-c. Chang, and P.-c. Lin, "Detecting Web-Based Botnets Using Bot Communication Traffic Features", *Security and Communication Networks*, vol. 2017, 2017.

[39] A. K. Sood, R. J. Enbody, and R. Bansal, "Dissecting SpyEye-Understanding the design of third generation botnets", *Computer Networks*, vol. 57, no. 2, pp. 436–450, 2013, ISSN: 13891286. DOI: 10.1016/j.comnet.2012.06.021. arXiv: 1005.3014. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.06.021.

[40] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the analysis of the Zeus botnet crimeware toolkit", *PST 2010: 2010 8th International Conference on Privacy, Security and Trust*, pp. 31–38, 2010, ISSN: 2274-2042. DOI: 10.1109/PST.2010.5593240. arXiv: 1512.08546.

[41] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The Rise of Social Botnets: Attacks and Countermeasures", *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1–14, 2016, ISSN: 1545-5971. DOI: 10.1109/TDSC.2016.2641441. arXiv: 1603.02714. [Online]. Available: http://arxiv.org/abs/1603.02714.

[42] J. Nazario, *Twitter-based Botnet Command Channel*, 2013. [Online]. Available: https://www.arbornetworks.com/blog/asert/twitter-based-botnet-command-channel/ (visited on 02/22/2018).

[43] L. Zeltser, *When Bots Use Social Media for Command and Control*, 2015. [Online]. Available: https://zeltser.com/bots-command-and-control-via-social-media/ (visited on 02/22/2018).

[44] T. M. Americas, "DDoS Attacks, New DDoS Taxonomy and Mitigation Solutions – A Survey", *International conference on Signal Processing, Communication, Power and Embedded System (SCOPES)-2016*, no. 2014, pp. 793–798, 2016.

[45] Kaspersky Lab, "Kaspersky ddos protection", p. 7, 2014.

[46] INCAPSULA, *DDoS Attack Types & Mitigation Methods*. [Online]. Available: https://www.incapsula.com/ddos/ddos-attacks/ (visited on 02/23/2018).

[47] C. Patrikakis, M. Masikos, and O. Zouraraki, "Distributed Denial of Service Attacks - The Internet Protocol Journal", *Cisco*, vol. 7, no. 4, 2004. [Online]. Available: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-30/dos-attacks.

html%20http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-30/dos-attacks.html.

[48] J. Postel, "User Datagram Protocol", Tech. Rep., 1980. [Online]. Available: https://tools.ietf.org/html/rfc768.

[49] ——, "Internet Control Message Protocol", Tech. Rep., 1981, p. 20. [Online]. Available: https://tools.ietf.org/html/rfc792.

[50] CORERO, *Definition of DDoS, Attack Types & Characteristics Glossary | Corero*. [Online]. Available: https://www.corero.com/resources/glossary.html%7B%5C#%7DICMP%20Flood%20https://www.corero.com/resources/glossary.html%7B%5C#%7DSlow%20Read%20Attack (visited on 02/26/2018).

[51] CloudFlare, *DNS Amplification | DDoS Attack Glossary | Incapsula*. [Online]. Available: https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/ (visited on 02/26/2018).

[52] M. Majkowski, *Memcrashed - Major amplification attacks from UDP port 11211*, 2018. [Online]. Available: https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/ (visited on 03/12/2018).

[53] Shodan, *Public Memcached Servers - Shodan*, 2018. [Online]. Available: https://www.shodan.io/report/poJtt1i9 (visited on 03/12/2018).

[54] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks", *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013, ISSN: 1553877X. DOI: 10.1109/SURV.2013.031413.00127.

[55] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations", Tech. Rep., 2007, p. 19. DOI: 10.17487/rfc4987. arXiv: arXiv:1011.1669v3. [Online]. Available: https://tools.ietf.org/html/rfc4987%20https://www.rfc-editor.org/info/rfc4987.

[56] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art", *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004, ISSN: 13891286. DOI: 10.1016/j.comnet.2003.10.003.

[57] W. M. Eddy, "Defenses Against TCP SYN Flooding Attacks", *The Internet Protocol Journal*, vol. 9, no. 4, pp. 2–16, 2006, ISSN: 1554-6578. DOI: 10.1097/NEN.0000000000000104. [Online]. Available: http://www-kiv.zcu.cz/%7B~%7Dledvina/DHT/ipj%7B%5C_%7D9-4.pdf.

[58] D. Bekerman and D. Breslaw, *How Mirai Uses STOMP Protocol to Launch DDoS Floods*, 2016. [Online]. Available: https://www.incapsula.com/blog/mirai-stomp-protocol-ddos.html (visited on 03/05/2018).

[59] Cloudflare, *Ping of Death DDoS attack | Cloudflare*. [Online]. Available: https://www.cloudflare.com/learning/ddos/ping-of-death-ddos-attack/ (visited on 02/27/2018).

[60] ——, *Smurf DDoS Attack | Cloudflare*. [Online]. Available: https://www.cloudflare.com/learning/ddos/smurf-ddos-attack/ (visited on 02/27/2018).

[61] J. Postel, "Internet Protocol", Tech. Rep., 1981, p. 45. [Online]. Available: https://tools.ietf.org/html/rfc791.

[62] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)", Tech. Rep., 2000, pp. 1–8. DOI: 10.17487/rfc2784. arXiv: arXiv:1011.1669v3. [Online]. Available: https://www.rfc-editor.org/info/rfc2784.

[63] B. Krebs, *KrebsOnSecurity hit with record DDoS*, 2016. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (visited on 02/28/2018).

[64] Radware, "ERT Threat Alert Reaper Botnet", Tech. Rep., 2017, pp. 1–4.

[65] Y. Xie and S.-z. Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites", *IEEE/ACMTRANSACTIONS ON NETWORKING*, vol. 17, no. 1, pp. 15–25, 2009. DOI: 10.1109/TNET.2008.925628.

[66] Radware, *The ultimate guide to everything you need to know about DDoS attacks*. 2015, p. 44.

[67] T. Yatagai, T. Isohara, and I. Sasase, "Detection of HTTP-GET flood attack based on analysis of page access behavior", *IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing - Proceedings*, pp. 232–235, 2007. DOI: 10.1109/PACRIM.2007.4313218.

[68] Akamai, "DNS Reflection, Amplification and DNS Water-torture", 2017.

[69] ——, "akamai's [state of the internet] / security Q1 2017 report", Tech. Rep., 2017, p. 26.

[70] T. Yoshida, K. Kawakami, R. Kobayashi, M. Kato, M. Okada, and H. Kishimoto, "Detection and Filtering System for DNS Water Torture Attacks Relying Only on Domain Name Information", *Journal of Information Processing*, vol. 25, pp. 854–865, 2017. DOI: 10.2197/ipsjjip.25.854. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjjip/25/0/25%7B%5C_%7D854/%7B%5C_%7Darticle.

[71] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source", *Proceedings - International Conference on Network Protocols, ICNP*, pp. 312–321, 2008, ISSN: 10921648. DOI: 10.1109/ICNP.2002.1181418.

[72] Imperva, "Attackers Use DDoS Pulses to Pin Down Multiple Targets , Send Shock Waves Through Hybrids", 2018. [Online]. Available: https://lp.incapsula.com/rs/804-TEY-921/images/Pulse-wave-attacks-and-their-impact-on-hybrid-mitigation-solutions.pdf.

[73] C. Cimpanu, *Pulse Wave - New DDoS Assault Pattern Discovered*, 2017. [Online]. Available: https://www.bleepingcomputer.com/news/security/pulse-wave-new-ddos-assault-pattern-discovered/ (visited on 07/17/2018).

[74] J. Vijayan, *'Pulse Wave' DDoS Attacks Emerge As New Threat*, 2017. [Online]. Available: https://www.darkreading.com/attacks-breaches/pulse-wave-ddos-attacks-emerge-as-new-threat-/d/d-id/1329657? (visited on 07/17/2018).

[75] G. Yaltirakli, *Slowloris*, 2018. [Online]. Available: https://github.com/gkbrk/slowloris.

[76] Division CERT, *1996 CERT Advisories*. 2017, p. 170, ISBN: 1565921240.

[77] ——, "1998 CERT Advisories", Tech. Rep., 2017, p. 78.

[78] P. Ferguson and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, 1998. [Online]. Available: https://tools.ietf.org/pdf/rfc2267.pdf.

[79] D. Dittrich, *the Tribe Flood Network Distributed Denial of Service Attack Tool*, 1999. [Online]. Available: https://staff.washington.edu/dittrich/misc/trinoo.analysis.txt (visited on 03/06/2018).

[80] BBC News, *BBC News | SCI/TECH | Yahoo brought to standstill*, 2000. [Online]. Available: http://news.bbc.co.uk/2/hi/science/nature/635048.stm (visited on 03/06/2018).

[81] L. Garber, "Denial-of-Service Attacks Rip the Internet", vol. 33, p. 6, 2000. DOI: 10.1109/MC.2000.839316.

[82] M. Eid, "Cyber-Terrorism and Ethical Journalism: A Need for Rationalism.", in *Ethical Impact of Technological Advancements and Applications in Society*, October 2010, Ottawa, 2016, p. 25. DOI: 10.4018/jte.2010100101.

[83] R. Naraine, *Massive DDoS Attack Hit DNS Root Servers*, 2002. [Online]. Available: http://www.cs.cornell.edu/people/egs/beehive/rootattack.html (visited on 03/06/2018).

[84] Y. Zhang, Y. Xiao, K. Ghaboosi, J. Zhang, and H. Deng, "A survey of cyber crimes", *Security and Communication Networks*, vol. 5, no. 4, pp. 422–437, Apr. 2012, ISSN: 19390114. DOI: 10.1002/sec.331. [Online]. Available: http://doi.wiley.com/10.1002/sec.331.

[85] M. Lesk, "The New Front Line: Estonia under Cyberassault", *IEEE Security and Privacy*, vol. 5, no. 4, p. 4, 2007. DOI: 10.1109/MSP.2007.98.

[86] C. Enterprises, *Internet Group Anonymous Declares "War on Scientology"*, Clearwater, 2008. [Online]. Available: https://prlog.org/10046797.

[87] S. Mansfield-Devine, "Anonymous: Serious threat or mere annoyance?", *Network Security*, vol. 2011, no. 1, pp. 4–10, 2011, ISSN: 13534858. DOI: 10.1016/S1353-4858(11)70004-6. [Online]. Available: http://dx.doi.org/10.1016/S1353-4858(11)70004-6.

[88] DAILY MAIL REPORTER, *'Anonymous' hackers hit Playstation and Sony websites in revenge for lawsuit | Daily Mail Online*, 2011. [Online]. Available: http://www.dailymail.co.uk/sciencetech/article-1373621/Anonymous-hackers-hit-Playstation-Sony-websites-revenge-lawsuit.html?ito=feeds-newsxml (visited on 03/08/2018).

[89] N. Anderson, *"Anonymous" attacks Sony to protest PS3 hacker lawsuit*, 2011. [Online]. Available: https://arstechnica.com/tech-policy/2011/04/anonymous-attacks-sony-to-protest-ps3-hacker-lawsuit/ (visited on 03/08/2018).

[90] DAILY MAIL REPORTER, *Sony executives bow in apology to 77m PlayStation users over hacked accounts | Daily Mail Online*, 2011. [Online]. Available: http://www.dailymail.co.uk/sciencetech/article-1382388/Sony-executives-bow-apology-77m-PlayStation-users-hacked-accounts.html (visited on 03/08/2018).

[91] CNN, *Anonymous strikes back after feds shut piracy hub Megaupload - CNN*, 2012. [Online]. Available: https://edition.cnn.com/2012/01/19/business/megaupload-shutdown/index.html (visited on 03/08/2018).

[92] M. Sauter, "'LOIC Will Tear Us Apart": The Impact of Tool Design and Media Portrayals in the Success of Activist DDOS Attacks", *American Behavioral Scientist*, vol. 57, no. 7, pp. 983–1007, 2013, ISSN: 00027642. DOI: 10.1177/0002764213479370.

[93] M. Bing, *The Lizard Brain of LizardStresser*, 2016. [Online]. Available: https://www.arbornetworks.com/blog/asert/lizard-brain-lizardstresser/ (visited on 03/09/2018).

[94] Cloudbric, *Who's Behind DDoS Attacks and How Can You Protect Your Website?*, 2015. [Online]. Available: https://www.cloudbric.com/blog/2015/09/whos-behind-ddos-attacks-and-how-can-you-protect-your-website/%20http://blog.cloudbric.com/2015/09/whos-behind-ddos-attacks-and-how-can.html (visited on 03/09/2018).

[95] U. CERT, *Heightened DDoS Threat Poised by Mirai and Other Botnets*, 2016. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA16-288A (visited on 03/10/2018).

[96] S. Hilton, "Dyn Analysis Summary Of Friday October 21 Attack | Dyn Blog", *Dyn*, pp. 20–22, 2016. [Online]. Available: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/%20https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/%7B%5C%%7D0Ahttp://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.

[97] A. Coggine, *Bitfinex Undergoing DDOS Attack, IOTA Wallets Temporarily Unavailable*, 2017. [Online]. Available: https://cointelegraph.com/news/bitfinex-undergoing-ddos-attack-iota-wallets-temporarily-unavailable (visited on 03/12/2018).

[98] B. Akolkar, *DDoS Attack Pulls Down Bitcoin Gold Website – CoinSpeaker*, 2017. [Online]. Available: https://www.coinspeaker.com/2017/10/24/ddos-attack-pulls-bitcoin-gold-website/ (visited on 03/12/2018).

[99] Radware, *OpCatalonia*, 2017. [Online]. Available: https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/opcatalonia/ (visited on 03/10/2018).

[100] Skottler, *February 28th DDoS Incident Report | GitHub Engineering*, 2018. [Online]. Available: https://githubengineering.com/ddos-incident-report/ (visited on 03/12/2018).

[101] Akamai SIRT Alerts, *memcached, now with extortion! - The Akamai Blog*, 2018. [Online]. Available: https://blogs.akamai.com/2018/03/memcached-now-with-extortion.html (visited on 03/12/2018).

[102] C. Morales, *NETSCOUT Arbor Confirms 1.7 Tbps DDoS Attack; The Terabit Attack Era Is Upon Us*, 2018. [Online]. Available: https://www.arbornetworks.com/blog/asert/netscout-arbor-confirms-1-7-tbps-ddos-attack-terabit-attack-era-upon-us/ (visited on 03/12/2018).

[103] C. Herbeger, Z. Gadot, Y. Ben-Erza, and O. Ofer, "Global Application & Network Security Report 2014-2015", Tech. Rep., 2015.

[104] G. Loukas and G. Oke, "Protection against Denial of Service Attacks: A Survey", *The Computer Journal*, p. 19, 2009. DOI: 10.1093/comjnl/bxh000.

[105] Arbor Networks, "20 YEARS OF DDoS ATTACKS", Tech. Rep., 2016, p. 700.

[106] M. Robinson, J. Mirkovic, M. Schnaider, S. Michel, and P. Reiher, "Challenges and Principles of DDoS Defense", no. 418, 2003.

[107] Cisco Systems, *WAN and Application Optimization Solution Guide Cisco Validated Design*, August. 2008, p. 238.

[108] M. Perry and T. Margoni, "Legal Consequences of Packet Inspection", *SSRN Electronic Journal*, p. 4, 2012.

[109] European Union, *EUR-Lex - 32002L0058 - EN*, 2002. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:en:HTML%20http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:HTML (visited on 03/19/2018).

[110] A. Cooper, "What your broadband provider knows about your web use deep packet inspection and communications laws and policies : hearing before the Subcommittee on Telecommunications and the Internet of the Committee on Energy and Commerce, House of Representatives, O", vol. 277, no. 1984, 1 online resource (v, 132 p.) ill. 2008. [Online]. Available: http://purl.fdlp.gov/GPO/gpo4088.

[111] D. Mahajan, "Distinguishing DDos Attack and Flash Event using Real- World Datasets with Entropy as an Evaluation Metric", *Machine Intelligence and Research Advancement (ICMIRA), 2013 International Conference on*, p. 5, 2013. DOI: 10.1109/ICMIRA.2013.24.

[112] P. Salvador and A. Nogueira, *A Pratical Approach to Corporate Networks Engineering*. River Publishers, 2013, p. 374, ISBN: 9788792982094.

[113] J. Mirković, G. Prier, and P. Reiher, "Source-end DDoS defense", *Proceedings - 2nd IEEE International Symposium on Network Computing and Applications, NCA 2003*, pp. 171–178, 2003. DOI: 10.1109/NCA.2003.1201153.

[114] 18 U.S. Code § 1030, *18 U.S. Code § 1030 - Fraud and related activity in connection with computers | US Law | LII / Legal Information Institute*, 1986. [Online]. Available: https://www.law.cornell.edu/uscode/text/18/1030 (visited on 04/03/2018).

[115] Crown, "Computer Misuse Act 1990", in *Computer Misuse Act 1990*, 2008, p. 16. DOI: 10.1016/0267-3649(90)90080-U. [Online]. Available: http://www.legislation.gov.uk/ukpga/1990/18/pdfs/ukpga%7B%5C_%7D19900018%7B%5C_%7Den.pdf.

[116] H. M. Jarrett and M. W. Bailie, *Prosecuting Computer Crimes*. Washington, DC: Office of Legal Education Executive Office for United States Attorneys, 2010, p. 213.

[117] S. Oh and K. Lee, "The need for specific penalties for hacking in criminal law", *Scientific World Journal*, vol. 2014, 2014, ISSN: 1537744X. DOI: 10.1155/2014/736738.

[118] Harvard University, *Regulations Concerning the Use of University Resources*. [Online]. Available: http://static.fas.harvard.edu/registrar/ugrad%7B%5C_%7Dhandbook/current/chapter5/regulations%7B%5C_%7Duniv%7B%5C_%7Dresources.html (visited on 04/03/2018).

[119] University of Oxford, *Statutes and Regulations: Regulations Relating to the use of Information Technology Facilities*, 2016. [Online]. Available: http://www.admin.ox.ac.uk/statutes/regulations/196-052.shtml (visited on 04/03/2018).

[120] J. Toit, *Active vs. Passive network monitoring: an infographic – Iris Network Systems*, 2016. [Online]. Available: https://www.irisns.com/active-vs-passive-network-monitoring-an-infographic/ (visited on 07/11/2018).

[121] C. Robitaille, *Network Performance: Active Monitoring? Passive Monitoring? How About Both?! - Accedian*, 2018. [Online]. Available: https://accedian.com/blog/performance-monitoring/network-performance-active-passive-monitoring/ (visited on 07/11/2018).

[122] N. L. M. V. Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks", *IEEE Network Operations and Management Symposium (NOMS)*, p. 8, 2014. DOI: 10.1109/NOMS.2014.6838228. [Online]. Available: https://ieeexplore.ieee.org/document/6838228/.

[123] H. T. Work, O. S. Types, S. Ratios, O. Speeds, P. Budgets, L. Loss, and B. Technology, "Understanding Network TAPs – The First Step to Visibility", pp. 1–9, 2017.

[124] L. H. Yeo, X. Che, and S. Lakkaraju, "Understanding Modern Intrusion Detection Systems: A Survey", 2017. arXiv: 1708.07174. [Online]. Available: http://arxiv.org/abs/1708.07174.

[125] M. Sabourin and A. Mitiche, "Optical character recognition by a neural network", *Neural Networks*, vol. 5, no. 5, pp. 843–852, 1992, ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80144-3.

[126] K. Shameer, K. W. Johnson, B. S. Glicksberg, J. T. Dudley, and P. P. Sengupta, "Machine learning in cardiovascular medicine: are we there yet?", *Heart*, heartjnl–2017–311 198, 2018, ISSN: 1355-6037. DOI: 10.1136/heartjnl-2017-311198. [Online]. Available: http://heart.bmj.com/lookup/doi/10.1136/heartjnl-2017-311198.

[127] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey", *Digital Communications and Networks*, 2017, ISSN: 23528648. DOI: 10.1016/j.dcan.2017.10.002. arXiv: 1802.06305. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S235286481730247X.

[128] S. Devi and T. Neetha, "Machine Learning based traffic congestion prediction in a IoT based Smart City", *International Research Journal of Engineering and Technology (IRJET)*, pp. 3442–3445, 2017.

[129] B. Intelligent and S. Aur, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*, ISBN: 9781491962299.

[130] T. M. Mitchell, *Machine Learning*. 1997, p. 414, ISBN: 0070428077.

[131] B. Marr, *What Is The Difference Between Artificial Intelligence And Machine Learning?*, 2016. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/%7B%5C#%7D2936e4b32742%20https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-ma (visited on 03/26/2018).

[132] M. Copeland, *The Difference Between AI, Machine Learning, and Deep Learning?*, 2016. [Online]. Available: https://www.intel.com/content/www/us/en/analytics/ai-luminary-reza-zadeh-video.html%20https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/ (visited on 03/26/2018).

[133] Y. Zhang, "Speech Recognition Using Deep Learning Algorithms", pp. 1–5, 2013.

[134] J. Padmanabhan, M. Jose, and J. Premkumar, "Machine Learning in Automatic Speech Recognition: A Survey", *IETE Technical Review*, no. March, pp. 37–41, 2015. DOI: 10.1080/02564602.2015.1010611.

[135] S. Kuiper, "Introduction to Multiple Regression: How Much Is Your Car Worth?", *Journal of Statistics Education*, vol. 16, no. November, p. 19, 2008. DOI: 10.1080/10691898.2008.11889579.

[136] R. Sunil, "Essentials of Machine Learning Algorithms ( with Python and R Codes )", *20.08.2015*, vol. 20, pp. 1–15, 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/%20https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/.

[137] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, "Zero-Shot Learning Through Cross-Modal Transfer", *Advances in Neural Information Processing Systems 26*, pp. 1–7, 2013. arXiv: arXiv:1301.3666v2.

[138] MIT Tech Review, *This Factory Robot Learns a New Job Overnight*, 2016. [Online]. Available: https://www.technologyreview.com/s/601045/this-factory-robot-learns-a-new-job-overnight/ (visited on 03/28/2018).

[139] D. C. Hogg, *Artificial Intelligence*. 1996, pp. 183–227, ISBN: 9780121619640. DOI: 10.1016/B978-012161964-0/50009-1. arXiv: arXiv:1011.1669v3. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780121619640500091.

[140] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation", *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*,

pp. 26–33, 2001, ISSN: 00043702. DOI: 10.3115/1073012.1073017. arXiv: arXiv:1011.1669v3. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1073012.1073017.

[141] M. Thoma, "Analysis and Optimization of Convolutional Neural Network Architectures", PhD thesis, 2017. arXiv: 1707.09725. [Online]. Available: http://arxiv.org/abs/1707.09725.

[142] R. Kaplan, D. Chambers, and R. Glasgow, *Big Data and Large Sample Size: A Cautionary Note on the Potential for Bias*, 2014. DOI: 10.1111/cts.12178.

[143] D. M. Hawkins, "The Problem of Overfitting", *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004, ISSN: 00952338. DOI: 10.1021/ci0342472.

[144] T. Dietterich, "Overfitting and undercomputing in machine learning", *ACM Computing Surveys*, vol. 27, no. 3, pp. 326–327, 1995, ISSN: 03600300. DOI: 10.1145/212094.212114. [Online]. Available: http://portal.acm.org/citation.cfm?doid=212094.212114.

[145] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, "Learning From Data", *Learning from Data*, vol. 21, no. 4, pp. 479–481, 2010, ISSN: 1044-3983. DOI: 10.1097/EDE.0b013e3181e13328.

[146] D. H. Wolpert, "The Existence of A Priori Distinctions Between Learning Algorithms", *Neural Computation*, vol. 8, no. 7, pp. 1391–1420, 1996, ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.7.1391. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1996.8.7.1391.

[147] T. M. Gil and M. Poletto, "MULTOPS : a data-structure for bandwidth attack detection", *Proceedings of the 10 th USENIX Security Symposium*, p. 3, 2001.

[148] J. Mirkovic and P. Reiher, "D-WARD: A source-end defense against flooding denial-of-service attacks", *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 216–232, 2005, ISSN: 15455971. DOI: 10.1109/TDSC.2005.35.

[149] V. Sekar and N. Duffield, "LADS: Large-scale Automated DDoS Detection System.", *USENIX Annual Technical Conference, General Track*, pp. 171–184, 2006. DOI: 10.1.1.128.4626. [Online]. Available: https://www.usenix.org/event/usenix06/tech/full%7B%5C_%7Dpapers/sekar/sekar%7B%5C_%7Dhtml/.

[150] I. Petiz, P. Salvador, A. Nogueira, and E. Rocha, "Detecting DDoS attacks at the source using multiscaling analysis", *2014 16th International Telecommunications Network Strategy and Planning Symposium, Networks 2014*, 2014. DOI: 10.1109/NETWKS.2014.6959267.

[151] M. Zekri, S. E. Kafhali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments", *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, pp. 1–7, 2017. DOI: 10.1109/CloudTech.2017.8284731. [Online]. Available: http://ieeexplore.ieee.org/document/8284731/.

[152] B. HSSINA, A. MERBOUHA, H. EZZIKOURI, and M. ERRITALI, "A comparative study of decision tree ID3 and C4.5", *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, pp. 13–19, 2014, ISSN: 2158107X. DOI: 10.14569/SpecialIssue.2014.040203. [Online]. Available: http://thesai.org/Publications/ViewPaper?Volume=4%7B%5C&%7DIssue=2%7B%5C&%7DCode=SpecialIssue%7B%5C&%7DSerialNo=3.

[153] S. Salvatore, *hping3 | Penetration Testing Tools*, 2014. [Online]. Available: https://tools.kali.org/information-gathering/hping3 (visited on 08/17/2018).

[154] IT Central Station, "DDoS Buyer's Guide and Reviews July 2018", p. 32, 2018.

[155] K. Armistead, *Cisco Stealthwatch*, 2018.

[156] S. learn, *Kernel PCA — scikit-learn 0.19.2 documentation*, 2018. [Online]. Available: http://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/decomposition/plot%7B%5C_%7Dkernel%7B%5C_%7Dpca.html (visited on 08/14/2018).

[157] L. Name, F. Name, O. Training, P. Training, C. Darin, R. O. Training, M. Kimberly, G. Deepa, E. Board, E. Principal, I. Primary, F. Systems, E. B. Study, and N. Co-investigator, *Introduction to Machine Learning*, 1. 2014, pp. 1–5, ISBN: 9780874216561. DOI: 10.1007/s13398-014-0173-7.2.

[158] M. Learning, "A Complete Tutorial on Tree Based Modeling from Scratch ( in R & Python )", vol. 20, pp. 1–55, 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/%7B%5C#%7Dfour.

[159] B. Gorman, *A Kaggle Master Explains Gradient Boosting | No Free Hunch*, 2017. [Online]. Available: http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/ (visited on 08/02/2018).

[160] I. Reinstein, *XGBoost, a Top Machine Learning Method on Kaggle, Explained*, 2017. [Online]. Available: https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html (visited on 08/02/2018).

[161] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection", *Signal Processing*, vol. 99, pp. 215–249, 2014, ISSN: 01651684. DOI: 10.1016/j.sigpro.2013.12.026. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2013.12.026.

[162] S. learn, *Receiver Operating Characteristic (ROC) — scikit-learn 0.19.2 documentation*, 2018. [Online]. Available: http://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/model%7B%5C_%7Dselection/plot%7B%5C_%7Droc.html (visited on 08/13/2018).

[163] Claudette, *10 Incredibly Effective Tips To Stop Wasting Time At Work - You Work Too Much*, 2017. [Online]. Available: https://youworktoomuch.com/10-incredibly-effective-tips-to-stop-wasting-time-at-work (visited on 07/18/2018).

[164] S. HEATHFIELD, *What Employers Do About Employees Surfing the Web at Work*, 2018. [Online]. Available: https://www.thebalancecareers.com/surfing-the-web-at-work-1919261 (visited on 07/18/2018).

[165] M. Prinzlau, *6 security risks of enterprises using cloud storage and file sharing apps | Digital Guardian*, 2016. [Online]. Available: https://digitalguardian.com/blog/6-security-risks-enterprises-using-cloud-storage-and-file-sharing-apps (visited on 09/05/2018).

[166] J. Gamblin, *Mirai-Source-Code*, 2016. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code.

[167] Wireshark, *LinkLayerDiscoveryProtocol - The Wireshark Wiki*. [Online]. Available: https://wiki.wireshark.org/Development/LibpcapFileFormat%20http://wiki.wireshark.org/LinkLayerDiscoveryProtocol (visited on 08/22/2018).

[168] C. Lucas, *Python network packet dissection frameworks shootout: Scapy vs Construct vs Hachoir vs Kaitai Struct – Adventures in Python*, 2017. [Online]. Available: https://pythonistac.wordpress.com/2017/03/09/python-network-packet-dissection-frameworks-shootout-scapy-vs-construct-vs-hachoir-vs-kaitai-struct/ (visited on 08/22/2018).